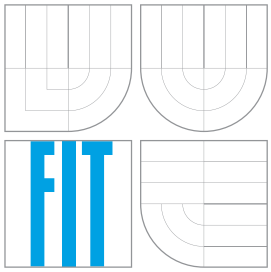
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

WEBOVÝ PŘEHRÁVAČ SE SLAJDY A PŘEPISEM

WEB BASE VIDEO PLAYER WITH SLIDES AND TRANSCRIPT

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. JAN KORIŤÁK

VEDOUCÍ PRÁCE
SUPERVISOR

IGOR SZŐKE, Ph.D.

BRNO 2016

Zadání diplomové práce/10617/2015/xkorit01

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav počítačové grafiky a multimédií Akademický rok 2015/2016

Zadání diplomové práce

Řešitel: **Koriťák Jan, Bc.**
Obor: Informační systémy
Téma: **Webový přehrávač videa se slajdy a přepisem**
Web Base Video Player with Slides and Transcript
Kategorie: Web

Pokyny:

1. Seznamte se s technologií HTML 5.
2. Seznamte se s rozhraním a funkcemi přehrávače SuperLectures.com. Nastudujte vlastnosti konkurenčních přehrávačů (slideslive.com, techtalks.com, atd.)
3. Navrhněte aplikaci s využitím HTML 5, která v sobě zapouzdří co nejvíce funkcí současného přehrávače videa a slajdů. Zaměřte se na pohodlné a intuitivní ovládání.
4. Implementujte navržený přehrávač. Měl by být robustní a rychlý.
5. Otestujte s pomocí uživatelů ergonomii a případně aplikaci upravte.
6. Zhodnoťte dosažené výsledky a navrhněte směry dalšího vývoje.
7. Vyrobte A2 plakátek a cca 30 vteřinové video prezentující výsledky vaší práce.

Literatura:

- Dle pokynů vedoucího

Při obhajobě semestrální části projektu je požadováno:

- Body 1, 2, 3 a část bodu 4 ze zadání.

Podrobné závazné pokyny pro vypracování diplomové práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva diplomové práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap, které byly vyřešeny v rámci dřívějších projektů (30 až 40% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Szőke Igor, Ing., Ph.D.**, UPGM FIT VUT
Datum zadání: 1. listopadu 2015
Datum odevzdání: 25. května 2016

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav počítačové grafiky a multimédií
612 66 Brno, Božetěchova 2

doc. Dr. Ing. Jan Černocký
vedoucí ústavu

Abstrakt

Tato práce se zabývá problematikou návrhu a implementace přehrávače videozáznamu s doprovodem slajdů ve webovém prostředí, jakožto JavaScriptového pluginu. Jelikož má výstup této práce sloužit jako náhrada již existujícího přehrávače, je první kapitola věnována jeho podrobné analýze. Další kapitoly se pak věnují samotnému návrhu, implementaci a testování nového přehrávače, na základě znalostí nabytých ve fázi analýzy.

Abstract

Content of this thesis is dedicated to problematics of design and implementation of a web base video player with slides as a JavaScript plugin. As the outcome of this thesis is supposed to replace an already existing player, the first chapter is dedicated to it's throughout analysis. Following chapters are the dedicated to desing, implementation and testing of the player, using information acquired in the phase of analysis.

Klíčová slova

Web, přehrávače, slajdy, vyhledávání, přepis

Keywords

Web, player, slides, search, transcript

Citace

KORIŤÁK, Jan. *Webový přehrávač se slajdy a přepisem.* Brno, 2016. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Szőke Igor.

Webový přehrávač se slajdy a přepisem

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana Ing. Igora Szőkeho, Ph.D. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Jan Koriťák
25. května 2016

Poděkování

Poděkování patří vedoucímu práce panu Ing. Igoru Szőkemu, Ph.D za návrh tématu a odbornou pomoc při řešení problému.

Obsah

1 Úvod **3**

2 Aktuální přehrávač **4**
2.1 Vzhled a funkce . . . 4
2.2 Technologie . . . 7
2.3 Komunikace se serverem . . . 9
2.3.1 Volání getLecture . . . 9
2.3.2 Volání getSubtitles . . . 10
2.3.3 Volání vyhledávacího query . . . 11
2.3.4 Volání logEvent . . . 11
2.4 Implemetace přehrávače . . . 11
2.5 Srovnání s konkurencí . . . 11

3 Návrh nového přehrávače **15**
3.1 Vzhled a funkce . . . 15
3.2 Podpora mobilních zařízení . . . 16
3.3 Fullscreen . . . 16
3.4 Architektura . . . 17
3.5 Technologie . . . 20
3.5.1 jQuery UI . . . 20
3.5.2 jQuery Mobile . . . 20
3.5.3 doT.js . . . 22

4 Implementace nového přehrávače **23**
4.1 jQuery Plugin . . . 23
4.2 Plugin . . . 24
4.2.1 Konfigurace . . . 24
4.2.2 Inicializace . . . 27
4.2.3 Jazykové mutace . . . 27
4.2.4 Sestavení HTML kódu . . . 27
4.2.5 Modální elementy . . . 29
4.2.6 Předání control-flow . . . 29
4.3 Player . . . 30
4.3.1 Detekce slajdů . . . 30
4.3.2 Fullscreen . . . 30
4.3.3 Zarovnání videa a slajdů . . . 32
4.3.4 Detekce aktivity uživatele . . . 33
4.4 VideoPlayer . . . 34

4.4.1 Inicializace . . . 35
4.4.2 Princip synchronizace . . . 35
4.5 SlidesPlayer . . . 36
4.5.1 Inicializace . . . 37
4.5.2 Sestavení přehledu . . . 37
4.5.3 Detekce dotykové obrazovky . . . 37
4.5.4 Centrování aktivního slajdu . . . 38
4.5.5 Preview slajdu . . . 39
4.5.6 Synchronizace a Přetáčení . . . 39
4.6 Transcript . . . 40
4.7 Search . . . 41
4.7.1 Sestavení vyhledávání . . . 41

5 Testování **43**
5.1 Ergonomie . . . 43
5.2 Kompatibilita . . . 45

6 Závěr **47**

Literatura **48**

Přílohy **50**
Seznam příloh . . . 51

A Objekt Lecture **52**

B jQuery Boilerplate **54**

C Konstrukce doT.js **55**

Kapitola 1

Úvod

Web Superlectures.com poskytuje služby v oblasti vytváření a následné správy videozáznamů z vybraných konferencí, přednášek, či workshopů různě po světě. Jelikož jsou v dnešní době k dispozici velmi kvalitní data projektory, využívá většina přednášejících na takovýchto událostech s výhodou doprovodu slajdů. Použití slajdů při přednášce je, z pravidla, výhodné pro obě strany - přednášejícího i posluchače.

Přednášející slajdy může použít jako osnovu či oporu při přednášení, díky které zajistí, že zmíní veškeré informace, které měl v úmyslu zmínit. Přednášející může dále na slajdy umístit složitěji strukturovaná data jako tabulky, nákresy, grafy, algoritmy, pokusy a podobně. Díky tomu nemusí tyto záležitosti na každé přednášce znovu reprodukovat, případně není nutno takové materiály nechávat kolovat, či situaci řešit úplně jinak.

Každý posluchač určitě doprovod slajdů také uvítá. Ne nadarmo se říká, že jeden obrázek vystačí za tisíc slov.

Z výše zmíněných důvodů vyplývá, že je vhodné mimo videozáznamu z přednášek rovněž uchovávat slajdy, které mnohdy bývají díky různé kvalitě videozáznamů nečitelné (pokud je záznam vůbec obsahuje) a není tudíž možné je flexibilně prohlížet.

Z tohoto důvodu web Superlectures.com poskytuje vlastní přehrávač, který spojuje dohromady přehrávání videozáznamu bok po boku s přiloženými slajdy. Přehrávač dokáže slajdy vzhledem k videozáznamu synchronizovat a podporuje širokou škálou rozšiřujících funkcí jako vzájemné přetáčení, vyhledávání fráze v audiu či náhled na transkript audio složky záznamu.

Díky pokroku webových technologií a zvýšení rychlosti bezdrátového připojení řada uživatelů v dnešní době přistupuje k webovým stránkám skrze mobilní zařízení, jejichž způsob interpretace webových stránek se mnohdy oproti desktopovým systémům velmi liší. Proto je nutno webové stránky optimalizovat pro podporu takových zařízení.

Obsahem této práce je postupný rozbor a analýza funkcí aktuálního přehrávače webu Superlectures.com a následný návrh, implementace a testování nového přehrávače, který by měl být schopen jednoduše nahradit přehrávač stávající stylem *Plug-and-play*.

Tento nový přehrávač by měl být, co do kódu, dobře strukturovaný a robustní, rovněž ale rychlý a co dobře optimalizovaný právě pro mobilní zařízení.

Kapitola 2

Aktuální přehrávač

Před započetím fáze návrhu nového přehrávače je nutno, v první řadě, dokonale porozumět přehrávači aktuálnímu. Zjistit, jak se přehrávač chová navenek (z pohledu uživatele). Konkrétně se jedná o získání informací o funkčním rozsahu, co leží v rámci funkcí nabízených přehrávačem a co je naopak za hranicí rozsahu. Dále získat informace, jak se chová za stanovených podmínek, jestli má nějaká důležitá omezení a podobně. Pokud je to možné, je ,v řadě druhé, vhodné shromáždit alespoň základní informace o tom, jak je přehrávač navržen a implementován z pohledu programátora, aby byl konečný proces nasazení náhrady přehrávače nejjednodušší možný.

Web Superletctures.com poskytuje dva různé webové přehrávače videa se slajdy. Přičemž je v nutno dodat, že oba přehrávače mají vestavěnou detekci situací, kdy záznam obsahuje přidružené slajdy a oba přehrávače dokáží náležitě nabídnout buďto pouze rozhraní pro přehrávání videa nebo kompletní rozhraní pro přehrávaní videa s doprovodem slajdů.

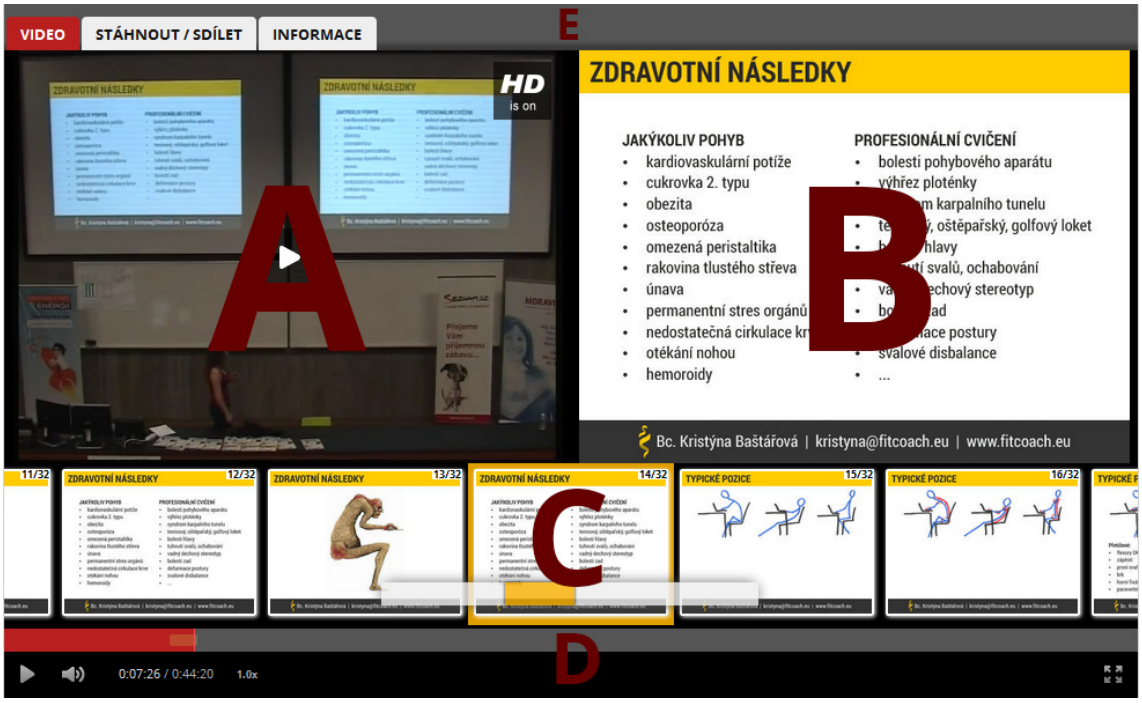
Prvním přehrávačem je starší přehrávač, jehož jádro tvoří široce distribuovaný multimediální a aplikační přehrávač Flash player. Okolo instance samotného Flash Playeru je vybudováno několik JavaScriptových modulů, které jako celek tvoří plnohodnotný přehrávač videa a slajdů. Hlavní modul zapouzdřuje funkčnost synchronizovaného přehrávače slajdů a několik dalších samostatných modulů zajišťuje další základní, ale i rozšiřující funkce pro pohodlnou práci se záznamem a přidruženými slajdy.

Cílem této práce je ale "refaktorizace"a návrh nového vzhledu pro novější přehrávač, jehož jádrem je videopřehrávač založený na technologii HTML5. V následujících sekcích bude provedena analýza rozložení vzhledu rozhraní přehrávače, společně s analýzou funkcí přehrávače, kterými disponuje v aktuální podobě. V první řadě se jedná o analýzu z pohledu běžného uživatele, následuje rozbor několika technologických a technických detailů z pohledu návrháře, resp. programátora.

Jelikož případ, kdy nejsou zobrazeny slajdy je z hlediska analýzy, jak rozložení vzhledu, tak funkcí, triviální, budou se další kapitoly věnovat rozboru přehrávače výhradně v případě, kdy slajdy zobrazeny jsou.

2.1 Vzhled a funkce

Aktuální přehrávač zapouzdřuje video a slajdy do jednoho společného elementu. Funkcím transkript a vyhledávání v audiu jsou vyhrazeny dva samostatné moduly, které se nachází mimo element přehrávače. Rozložení přehrávače lze intuitivně rozdělit do pěti segmentů. Tyto segmenty mohou být označeny např. následovně, písmeny abecedy A - E. Viz. 2.1.



Obrázek 2.1: Aktuální podoba přehrávače Superlectures.com

Video (Segment A) Levá část přehrávače je vyhrazena přehrávači videa. Poměr velikosti přehrávače videa ku velikosti slajdů je stanoven na 1:1. Přehrávač videa, v tomto případě, v rámci okna nabízí jedinou funkci, kterou je možnost přepínání mezi HD a SD verzí videozáznamu. Vlastnosti videopřehrávače dále plynou z konkrétní použité platformy. Videopřehrávač je sestaven s využitím platformy JWPlayer (viz. dále).

Slajdy (Segment B) V pravé části přehrávače jsou umístěny slajdy. Slajdy jsou implicitně synchronizovány s videozáznamem v segmentu A. Změna slajdu se projeví přetočením videozáznamu na odpovídající čas, stejně jako se přetočení videa projeví odpovídající synchronizací slajdů.

Přehled slajdů (Segment C) Pod přehrávači videa a slajdů se nachází horizontálně orientovaný kontejner, který obsahuje chronologicky seřazený přehled všech slajdů, které patří k běžícímu videozáznamu. Právě tento kontejner poskytuje rozhraní pro výběru slajdu. Mezi slajdy se dá pohodlně "listovat"pomocí horizontálního posuvníku. Listování je možno provádět několika způsoby. Běžným způsobem je využití zmíněného horizontálního posuvníku, nicméně kontejner reaguje i na použití kolečka uživatelovy myši. Na mobilních zařízeních je posuvník nahrazen klasickým HTML *scroll* panelem, který dokáže korektně reagovat na ovládání pomocí dotykových událostí. Panel se rovněž, po určité době nečinnosti, automaticky centruje na pozici aktuálně aktivního slajdu, který je takto periodicky zarovnáván na střed svého kontejneru a tudíž i obrazovky. Každý ze slajdů v rámci kontejneru je označen dvojicí čísel ve formátu *aktualni_slajd/celkovy_pocet_slajdu*. Slajd, který je aktuálně přehráván je viditelně odlišen od ostatních pomocí jednoduché změny barvy vnitřního okraje. Přejetím myši přes jakýkoli slajd (resp. událost *hover*) poskyuje funkci zaostření. Tedy maximalizaci daného slajdu do popředí tak, že překryje aktivní přehrávací plochu. Jinými slovy

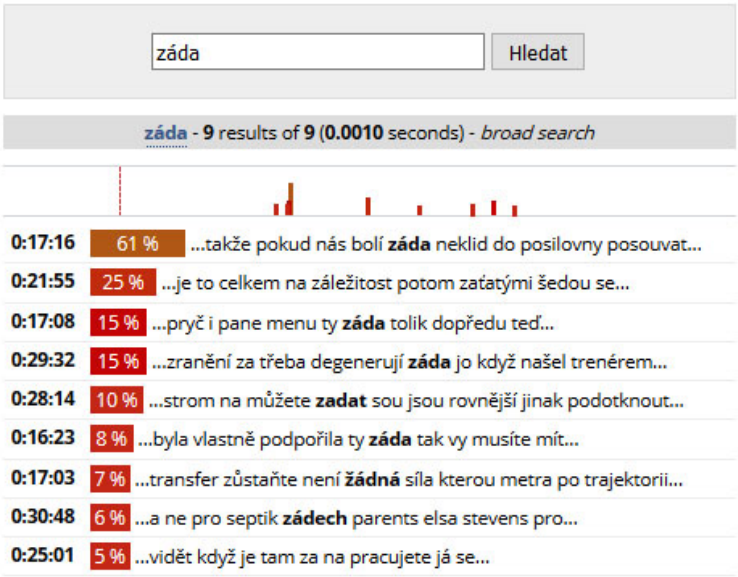
se jedná o modální element. Dále v textu bude tento segment odkazován jako *progress-bar* slajdů.

Ovládací panel (Segment D) Segment ovládacího panelu zahrnuje veškerá tlačítka pro ovládání videozáznamu a tlačítko přechodu přehrávače do režimu celé obrazovky, resp. *fullscreen*. Výčet tlačítek pro ovládání záznamu obsahuje tlačítko *Play/Pause*, pro přehrání/pozastavení záznamu. Tlačítko a posuvník pro ovládání hlasitosti audio složky záznamu, případně její úplné ztlumení. Přehled o průběhu času záznamu ve formátu *aktualni_cas/celkovy_cas* a tlačítko pro ovládání rychlosti přehrávání záznamu (povolené hodnoty jsou 1.0x - 1.5x).

Záložky (Segment E) Záložky v horní části přehrávače zapouzdřují rozšířenou funkcionalitu přehrávače. Základní zobrazení přehrávače se odehrává v první záložce. V pořadí třetí záložka nabízí náhled na informace o videozáznamu, jak už statistického rázu (počet přehrání, graf sledovanosti a další různá statistická data), tak např. informace o přednášejícím, abstrakt nebo motivaci přednášky. V pořadí druhá záložka pak umožňuje stažení záznamu ve formátu .mp4 nebo .webm, případně stažení pouze audio složky ve formátu .mp3 nebo stažení transkriptu v různých formátech. Dále pak dovoluje sdílení záznamu na sociálních sítích, případně poskytuje připravený kód pro *embeded* verzi záznamu.

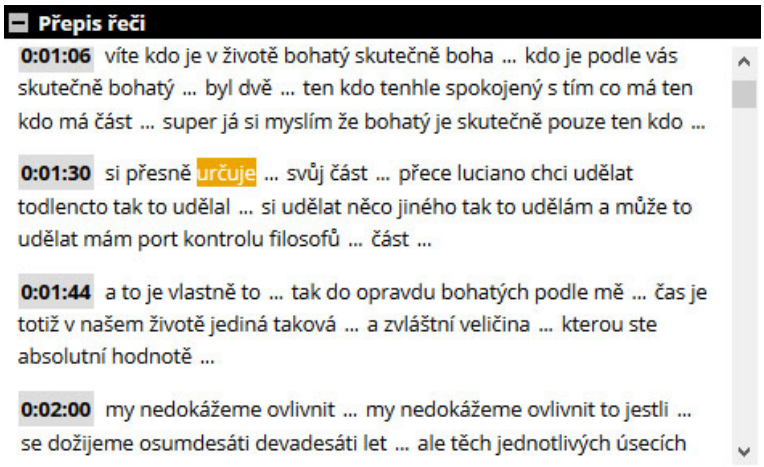
Funkce mimo přehrávač Jako segment F by se daly označit funkce mimo element přehrávače.

Jedná se o rozhraní pro vyhledávání fráze v audio záznamu, které poskytuje informace o tom, kolikrát se daná fráze v záznamu vyskytuje, k čemuž přidává přehlednou časovou osu s vyznačenými místy výskytu fráze, doprovázené označením přesnosti shody, vyjádřené v procentech. Mimo zmíněnou osu jsou všechny, do kontextu vložené, seřazené fráze, doprovázeny časovou známkou, zobrazeny v doprovodném seznamu. Samotné fráze pak fungují jako odkaz na čas záznamu jejich výskytu. Viz. následující obrázek 2.2.



Obrázek 2.2: Modul pro vyhledávání fráze v audio záznamu

Poslední funkcí, mimo nabídku podobných záznamů, je interaktivní náhled na transkript audio záznamu, který funguje velmi podobně jako, výše zmíněný, *progress-bar* slajdů. Jedná se o vertikálně orientovaný kontejner, který na každém svém "řádku"zobrazuje právě jednu frázi transkriptu. Transkript je rovněž aktivním prvkem, tedy je synchronizován vůči videozáznamu, a dokáže označit frázi, která je aktuálně podávána přednášejícím v rámci záznamu. I transkript poskytuje funkci centrování na aktivní frázi po určitě době nečinnosti. Vyobrazeno na obrázku 2.3.



Obrázek 2.3: Modul pro vyhledávání fráze v audio záznamu

Kontejnery slajdů a videa jsou v poměru hran 4:3 (šířka:výška). Kontejner videa nabývá poměru hran 4:3 i v situaci, kdy je video zaznamenáno v poměru 16:9. Slajdy jsou vždy 4:3.

2.2 Technologie

Aktuální přehrávač Superlectures.com je implementován pomocí dnes klasické trojice jazyků, užívaných v oblasti webových technologií, HTML5, CSS3 a JavaScriptu s využitím knihovny jQuery. Pro přehrávání videa je využit vestavěný přehrávač JWPlayer. Data ze serveru přehrávač přijímá buď ve formátu XML nebo JSON.

HTML Jazyk HTML5 je zatím posledním standardem z rodiny značkovacích jazyků HTML. Jedná se o esenci drtivé většiny webových stránek dneška. Jazyk HTML s každou verzí nabízí programátorovi širší repertoár funkcí, které poskytují pohodlné řešení pro problémy, které v minulosti bylo nutno řešit například s využitím JavaScriptu. Od zmíněné verze HTML5 jde především o funkce zabezpečující přímou podporu přehrávání multimédií v rámci prohlížeče. Dále například podpora funkce plátna, pro vykreslování vektorové grafiky nebo podpora pro spuštění aplikací, které nadále fungují i bez připojení k internetu [1].

CSS CSS je stylovací jazyk, který lze využít k definici stylu dokumentu napsaném, v jednom z jazyků, HTML, XHTML nebo XML. Velkou výhodou jazyka CSS je, že dovoluje oddělení definice stylu dokumentu od definice jeho struktury, což implikuje snadnou znovupoužitelnost vytvořených stylů a přehlednost samostatně definovaných pravidel.

V průběhu roku 2011 byl vydán zatím poslední standard CSS, který je označován jako CSS3 a zavádí celou řadu užitečných modulů.

V rámci této práce byly využity hlavně moduly pro *media-queries*, které umožňují programátorovi omezit definici stylů dokumentu sadou podmínek, při jejichž splnění (nebo naopak nesplnění) se styly patřičně přizpůsobí. Funkcionalita modulu *media-queries* se s výhodou využívá pro definici tzv. responzivního webu i pro mobilní zařízení.

Modulů je mnoho a nachází se v rozdílných stádiích vývoje a to jak, co se týče standardizace, tak co se týče podpory v různých prohlížečích. Z tohoto důvodu je stále nutné s některými CSS3 vlastnostmi pracovat v ne-standardní podobě s využitím tzv. *vendor-prefixů*.

Vendor prefix je souhrnné označení pro sadu předpon některých CSS3 vlastností, které jsou zatím nestandardizované nebo v experimentální fázi vývoje. Díky těmto prefixům vývojáři používající tyto nestandardní technologie nejsou závislí na změnách v jejich implementaci během standardizačního procesu. Vlastnost bez prefixu je z pravidla zavedena až po plné standardizaci [12].

V rámci této práce se pro úplnou podporu prohlížečů používají všechny čtyři standardní prefixy.

- -webkit- pro Chrome, Safari a Opera
- -moz- pro Firefox
- -o- pro starší verze Opery. Opera je dnes součástí webkit.
- -ms- pro Internet Explorer

JavaScript a jQuery Jazyk JavaScript je interpretovaný, multiplatformní, objektově orientovaný skriptovací jazyk, který se téměř výhradně požívá k přidání dynamického chování a interaktivity na straně webového prohlížeče.

K popularizaci JavaScriptu, mimo jiné, přispělo vydání rychlé, úsporné, ale funkčně velmi bohaté knihovny jQuery. Knihovna poskytuje pohodlné aplikační rozhraní, které zjednodušuje zápis pro např. pohodlnou práci s DOMem, zpracování událostí na straně prohlížeče, animace nebo práci s AJAXem.

AJAX Zkratkové slovo AJAX reprezentuje Asynchronní JavaScript a XML. Zavedení asynchronních funkcí do tvorby webových stránek rovněž přispívá ke zvýšení úrovně dynamiky. Asynchronní zpracování umožňuje v určitých situacích aktualizovat část dokumentu bez nutnosti znovunačítání celé stránky.

JWPlayer JWPlayer je velmi populární vestavěný videopřehrávač, vybudovaný na technologii HTML5. JWPlayer je *open-source* projekt, který podporuje širokou škálu cílových platforem, prohlížečů a formátů.

Společně s přechodem do verze 7 JWPlayer nabízí vývojářům aktualizované API jak pro JavaScript, tak v pro Android, v podobě Android SDK, tak pro iOS v podobě iOS SDK.

Od verze 7 je rovněž možné volitelně konfigurovat vzhled přehrávače skrze vestavěné atributy.

XML Je formát, který se používá k serializaci dat při přenosu mezi aplikacemi. Zápis XML se soustředí na popis struktury dat z hlediska věcného obsahu, nezabývá se popisem vzhledu dokumentu. Ten může být definován např. pomocí zmíněného jazyka CSS. Zpracování formátu XML je v dnešní době podporováno velkým množstvím různých nástrojů a programovacích jazyků. Výjimkou není ani zmíněný JavaScript.

JSON Alternativou XML je další serializační formát JSON. JSON, neboli JavaScriptová objektová notace, je odlehčený, snadno zapisovatelný i čitelný formát. Je založen na podmnožině programovacího jazyka JavaScript, standard ECMA-262 3rd edition. Je to textový, na jazyce nezávislý formát, využívající konvence jazyků z rodiny C, jazyka Java a dalších skriptovacích jazyků jako Perl a Python [9].

2.3 Komunikace se serverem

Sledováním síťové komunikace lze zjistit, že přehrávač Superlectures.com za svého běhu zasílá velké množství asynchronních požadavků na server. Ostatně, přehrávač tímto způsobem získává veškerá data, která následně prezentuje - a to ve formátu XML nebo ve formátu JSON. Všechny dotazy jsou směřovány na připravené API.

2.3.1 Volání getLecture

Prvním dotazem, který přehrávač na server zasílá je volání funkce API `getLecture`, která přijímá právě jeden parametr, který identifikuje daný záznam v rámci události. Skrze JSON-P *callback* server vrací poměrně rozsáhlý JSON záznam, který zapouzdřuje veškerá data konkrétního záznamu v serializované podobě.

Server dokáže tento záznam poskytnout i ve formátu XML (viz. příklad v příloze) a jelikož je slovní popis, díky terminologii, snazší sestavit pro formát XML, bude při popisu na záznam pohlíženo jako záznam ve formátu XML.

Kořenovým elementem XML záznamu je element `<data>`, který obsahuje následující elementy.

`<lecture>` Jediným přímým potomkem elementu `data` je element `lecture`. Je rodičem všech následujících elementů a funguje tedy pouze jako kontejner. Atributy elementu `lecture` poskytují základní informace o záznamu. Identifikátor videozáznamu a jeho kategorie, titulek videozáznamu, jazyk videozáznamu či URL adresu.

`<authors>` Rodičovský element záznamů o autorech přednášky. Každý autor je dále popsán elementem `<person>`, který poskytuje různé varianty kontaktu na danou osobu jako jsou odkazy na Facebook účet, Twitter účet, adresu emailové schránky, odkaz na osobní blog osoby a několik dalších forem kontaktu.

`<speakers>` Rodičovský element záznamů o podílejících se mluvčích. Každý mluvčí je rovněž popsán pomocí elementu `<person>`, který má stejnou strukturu jako v případě `<authors>`.

`<description>` Textový popis dané přednášky.

<recorded> a <datetime_recorded> Datum a čas, kdy se přednáška odehrávala.

<datetime_added> Datum a čas, kdy byla přednáška zpřístupněna.

<thumbnail> a <thumbnail_large> URL adresa úvodního obrázku přednášky, který se dále používá pro náhled.

<video> Obsahuje atribut *duration*, neboli délku trvání videozáznamu v sekundách, a několik synovských elementů s URL adresami videozáznamu v různých rozlišeních a formátech. Videozáznam je dostupný např. ve formátech .flv, .mp4, .webm.

<audio> Rovněž obsahuje atribut `duration`, neboli délku trvání audiozáznamu v sekundách (zpravidla odpovídá délce videozáznamu). Synovské elementy obsahují adresy audiozáznamu v různých formátech, např. .mp4 nebo .wf.

<subtitles> Pokud jsou dostupné titulky, informace o nich zahrnuje element `subtitles`.

<annotation> Anotace přednášky ve formátu textu.

<slides> Pomocí atributu `number_of_slides` poskytuje informaci o počtu dostupných slajdů. Každý potomek elementu `slides` reprezentuje právě jeden slajd. Každý slajd zná čas svého nástupu ve videozáznamu. Každý slajd je dostupný ve více variantách podle požadovaného rozlišení.

<pdf> Adresa pro stažení transkriptu ve formátu .pdf.

<epub> Adresa pro stažení transkriptu ve formátu .epub.

Další elementy Záznam obsahuje několik dalších, méně podstatných elementů. Např. potomci elementu `stats` a element `views` poskytují statistické informace o sledovanosti videozáznamu.

2.3.2 Volání getSubtitles

V pořadí druhým voláním na server je volání funkce API `getSubtitles`, která přijímá právě jeden parametr, kterým je identifikátor dané přednášky.

Pokud přednáška obsahuje externí titulky, metoda vrací JSON záznam, který obsahuje pole objektů, kde každý objekt odpovídá jedné "frázi"titulků.

Každý z těchto objektů má právě 3 atributy, které obsahuje znění dané fráze a časové známky jejího nástupu a konce v rámci videozáznamu.

Titulky se proto používají k sestavení tanskriptu.

2.3.3 Volání vyhledávacího query

Posledním voláním, kterým přehrávač získává data pro prezentaci souvisí s vyhledáváním fráze v audiozáznamu.

Základní parametr `q` určuje, jaká fráze se má vyhledat. Další argumenty určují např. v jakých zdrojích na serveru Superlectures.com má být vyhledáváno (řeč, slajdy, abstrakt, autor, apod.), tolerance na shodu vyhledávané fráze, kolik výsledků má být zasláno klientovi, případně v jakém pořadí.

Výsledkem je opět poměrně rozsáhlá XML/JSON stuktura, která zapouzdřuje nejen všechny nalezené výskyty fráze v daných zdrojích, ale i další informace o odeslaném query jako doba zpracování požadavku serverem, či doba vyhledávání.

2.3.4 Volání logEvent

Voláním, které se za dobu běhu přehrávače provede průměrně nejvíce krát je volání metody `logEvent`. Metoda `logEvent` má několik parametrů, mezi které , opět, patří identifikátor daného záznamu, identifikátor zdroje vzniku události (v tomto případě přehrávač `slplayer`) nebo atributy `action` a `label`, které blíže upřesňují událost, která nastala. `Action` určuje konkrétní událost (např. klik) a `label` dodatečně specifikuje, nad kterým elementem byla událost provedena (např. klik na tlačítko *Pause*).

Jedná se tedy o typickou "logovací"funkci, která sleduje a zaznamenává každou akci uživatele na webu a odesílá je na server Superlectures.com. Takto uchované statistiky mohou být déle využity k analýze.

2.4 Implemetace přehrávače

Náhledem do zdrojového kódu stránky lze vyčíst, že je přehráváč implementován jako plugin pro knihovnu jQuery pod identifikátorem `slplayer`. Následující úsek kódu je typickou konktrukcí používanou při volání jQuery pluginu.

Plugin `slplayer` dynamicky vytváří instanci přehrávače, kterou následně vkládá do objektu DOMu s identifikátorem `mediaplayer`.

```
$("#mediaplayer").slplayer({
    /* Atributy anonymního objektu jako parametry  přehrávače */
});
```

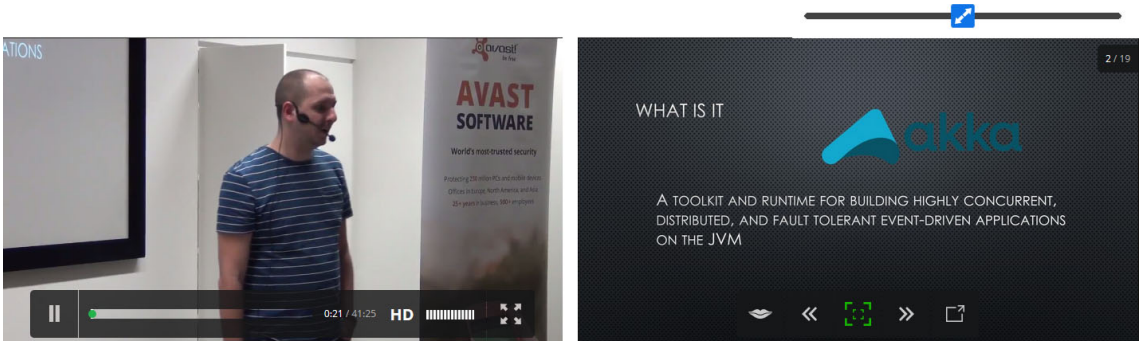
2.5 Srovnání s konkurencí

V současné době existuje nespočet webových projektů, které nabízejí služby pro sledování klasických videozáznamů. Existuje rovněž několik aktivních projektů, které poskytují služby, k nimž je zapotřebí přehrávač videa se synchronizovanými slajdy. Část těchto projektů disponuje vlastní implementací přehrávače, zatímco zbytek používá některý z volně dostupných přehrávačů. Hrstka z těchto webů pak tvoří přímou konkurenci webu Superlectures.com. Před návrhem a implementací nového přehrávače je výhodné provést průzkum konkurenčních přehrávačů, které mohou sloužit jako inspirace pro tvorbu přehrávače. Proto se právě analýze vlastností některých konkurenčních přehrávačů se věnují následující odstavce.

SlidesLive.com Přímou a nejbližší konkurencí webu Superlectures.com je český projekt, který v minulosti vystupoval pod názvem MetaTV, ale dnes vystupuje pod jménem Slideslive. Slideslive disponuje vlastním přehrávačem, který má v levé části okna, umístěn videozáznam a pravé části okna slajdy. Na rozdíl od přehrávače Superlectures obě okna disponují vlastním kontrolním panelem. Daný panel dovoluje kontrolovat odpovídající část přehrávače nezávisle na části druhé. Synchronizace neprobíhá okamžitě po přetočení videa nebo slajdu, nýbrž používá synchronizace ad-hoc, kterou zajišťují speciální tlačítka *Sync slides to Video* a *Rewind video to selected slide* 2.4.

Mimo základní zmíněné funkce poskytuje přehrávač Slideslive, v horní části umístěný, posuvník, který umožňuje dynamickou změnu poměru velikostí části přehrávače náležící videu, ku části náležící slajdům. Tato funkce může pomoci v různých situacích, kdy je výhodné zaostřit na video, resp. detailněji nahlédnout na slajdy.

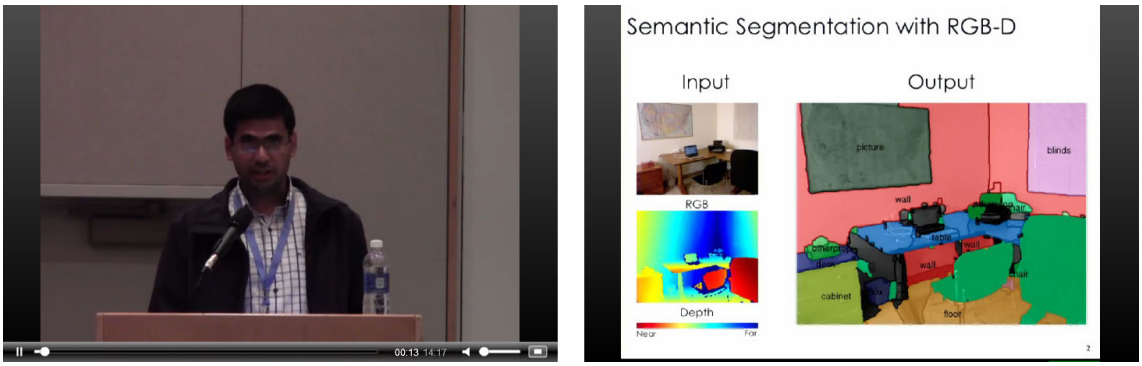
Přehrávač není optimalizován pro přehrávání na mobilních zařízeních.



Obrázek 2.4: Přehrávač SlidesLive.com

Techtalks.tv Další z projektů, který disponuje vlastním přehrávačem je Techtalks.tv. Přehrávač Techtalks je velmi jednoduchý. Levá část obrazovky je, klasicky, vyhrazena přehrávači videozáznamu a strana pravá slajdům. Přehrávač nestaví slajdy na stejnou úroveň jako videozáznam, ale bere slajdy pouze jako doprovod. Podporuje tedy pouze přetáčení videa, což se okamžitě projeví synchronizací slajdů. Opačný přístup není možný.

Zajímavou funkcí přehrávače Techtalks je, že dovoluje zvlášť aktivovat mód celé obrazovky pro videozáznam nebo slajdy, což je velmi podobná funkčnost, kterou nabízí rovněž zmíněný přehrávač Slideslive.



Obrázek 2.5: Přehrávač Techtalks.tv

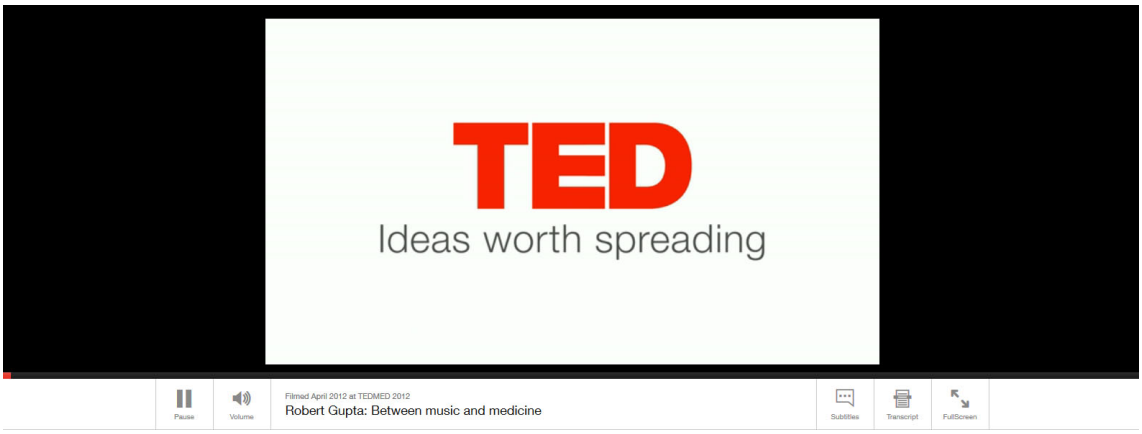
Videolectures.net Největší podobnost přehrávači Supelectures vykazuje přehrávač projektu Videolectures.net. Web videolectures.net je otevřeným repozitářem, který shromažďuje záznamy přednášek vybraných osobností z různých oblastí vědy, záznamy z různých konferencí, školení či workshopů. Přehrávač disponuje identickým rozložením segmentů s přehrávačem superlectures. Rovněž disponuje rozhraním pro náhled na strojově generovaný transkript po boku videozáznamu, kterému dokonce vyhrazuje samostatný modul přehrávače. Zmíněný přehrávač nabízí možnost zobrazení v několika různých *layoutech*.

Přehrávač není optimalizován pro přehrávání na mobilních zařízeních.

Obrázek 2.6: Přehrávač Videolectures.net

Ted.com "Ideas worth spreading"- Nápady, které stojí za šíření. To je slogan, kterým se identifikuje populární web s naučnými videi Ted.com. Přestože web Ted nedisponuje přehrávačem videa se slajdy a dokonce i zaměření webu se liší, minimálně jeden pohled na přehrávač je inspirující. Přehrávač je obalen v moderním, odlehčeném designu, s *full-resolution layoutem* a je velmi dobrou inspirací pro návrh atraktivního vzhledu přehrávače.

Přehrávač webu Ted.com je výborně optimalizován pro mobilní zařízení.



Obrázek 2.7: Přehrávač Ted.net

Z předchozích odstavců lze soudit, že aktuální podoba přehrávače v žádném směru nezaostává oproti konkurenčním produktům. Ba naopak, v mnoha ohledech je napřed. Přehrávač Superlectures.com je rozložením optimální, funkčně velmi bohatý a chodem stabilní a rychlý. Je zde několik záležitostí, které je možné přehrávači vytknout. Nicméně žádná z nich není existenčně důležitá.

První je poněkud neaktuální vzhled. Při návrhu nového vzhledu se lze inspirovat například právě vzhledem přehávače projektu Ted.com.

Druhou je řešení *fullscreen* módu, který nabízí pouze zvětšení přehrávače v rámci *viewportu*, nikoli v rámci celé obrazovky. Dnešní technologie nabízejí řešení tzv. *true fullscreen* nebo *native fullscreen* módu jak pro desktopová zařízení, tak dokonce pro velkou část mobilních zařízení.

Jako třetí záležitost lze vytknout nedokonalou podporu pro zobrazení přehrávače na mobilních zařízeních. Vzhled přehrávače počítá pouze s tzv. *landscape* orientací zařízení (zobrazení na šířku) a nemá přizpůsobený zvláštní *layout* pro polohu zobrazení portrétu, což je označení pro situaci, kdy je mobilní zařízení orientováno - klasicky - vertikálně. Dále pak přehrávač vykazuje nedostatky při přechodu mezi zmíněnými polohami v módu *fullscreen*, kdy se chová nekonzistentně při různých rozlišeních cílových zařízení.

Refaktorizace a vyřešení těchto nedostatků v novém produktu dostane přehrávač na vyšší úroveň. Hlavně v případě vyřešení dokonalé podpory pro zobrazení a přehrávání na mobilních zařízeních by takto celý web mohl získat z pohledu technologií konkurenční výhodu.

S využitím nově nabytých informací je nyní možné provést návrh nového přehrávače.

Kapitola 3

Návrh nového přehrávače

Informace nashromážděné v kapitole analýzy přehrávače Superlectures.com mohou nyní být využity jako vstup kapitoly návrhu. Tato kapitola se prezentuje návrh vzhledu nového přehrávače, jeho funkcí a architektury.

3.1 Vzhled a funkce

Navržený přehrávač v základním *non-fullscreen* módu lze vidět na následujícím obrázku 3.1.

Obrázek 3.1: Layout nového přehrávače

Nový návrh zachovává *layout* starého přehrávače, tudíž je opět možno označit jeho části písmeny abecedy A-E, totožně jako v kapitole analýzy. Také počítá s implementací veškerých funkcí starého přehrávače, včetně náhledu na transkript a vyhledávání fráze v audiu.

Do kontrolního panelu přibyla dvě tlačítka. Tlačítko pro změnu kamery a tlačítko pro změnu poměru velikostí přehrávače video ku velikosti slajdu. Obě tato tlačítka byla umístěna do pravé části kontrolního panelu, který obsahuje tlačítka funkcí, které ovlivňují obecné chování přehrávače.

Kromě této drobné změny nový layout počítá se vtažením modulu pro vyhledávání fráze v audiu a modulu pro práci s transkriptem, které byly v minulosti umístěny mimo přehrávač, dovnitř a to přidáním dvou dedikovaných záložek - "Vyhledávání"a "Transkript". Tato změna pomůže docílit odstranění závislosti vrstvy modelové na vrstvě prezentační a zároveň větší koheze v rámci prezentační vrstvy přehrávače.

3.2 Podpora mobilních zařízení

Nová implementace přehrávače, oproti implementaci staré, bude klást velký důraz na podporu mobilních zařízení a to hlavně v oblasti korektního zobrazení uvedeného *layoutu* v obou režimech - normální i *fullscreen* - a v oblasti korektního zpracování dotykových událostí na co nejširším spektru těch zařízení.

Řešení korektního zobrazení *layoutu* bude, v první řadě, prováděno za pomocí CSS *media-querries* a v řadě druhé, resp. v situacích, na které není CSS dostačující nebo pro jejich řešení není vhodné, bude využito JavaScriptu.

3.3 Fullscreen

Jak již bylo zmíněno, webové technologie dneška umožňují na některých platformách (dnes je to většina běžně dostupných platforem) využít tzv. nativního *fullscreen* módu. Tento nativní režim celé obrazovky dovoluje roztáhnout vybraný element (v tomto případě video a aktuální slajd) webu na celou obrazovku zařízení tak, že překryje všechny vrstvy pod sebou, tudíž i okno webového prohlížeče. Jinak řečeno, element vystoupí do popředí.

Nový přehrávač této funkce na podporovaných platformách využije, zatímco na platformách, kde tento režim podporován není, zachová klasický režim *fullsceen* v rámci *viewportu* okna prohlížeče, obdobně, jak je tomu u aktuálního přehrávače.

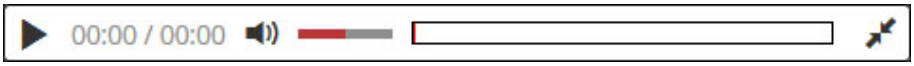
Nový přehrávač v režimu *fullscreen* již nebude vůbec zobrazovat záložky, které v současné podobě narušují uživatelský zážitek v režimu celé obrazovky. Dále přehrávač dozná rozšíření, které je cílené právě na uživatele, kteří přistupují k webu pomocí mobilních zařízení. Nový návrh počítá mimo klasického rozložení v režimu *landscape* s reorganizací *layoutu* přehrávače do speciálního módu v režimu portrét.

Z důvodu neúplné podpory nativního *fullscreenu* na všech platformách bude nový přehrávač nabízet dvě různé verze *layoutu* v režimu celé obrazovky (mimo tedy *portrait* a *landscape*).

Pokud se jedná o zařízení, na kterém je nativní režim celé obrazovky povolen, dojde ke skrytí, nejen záložek, ale rovněž celého kontrolního panelu, včetně *progress-baru* videa a přehledu slajdů. Klasický kontrolní panel bude nahrazen velmi odlehčenou verzí, která bude dále označována jako tzv. *lite* verze. Zmíněný odlehčený kontrolní panel bude obsahovat pouze základní podmnožinu ovládacích prvků přehrávače jako tlačítka *Play/Pause/Replay*, panel pro ovládání zvuku a minimalizovaný *progress-bar* videa pro možnost přetáčení videa i v režimu *fullscreen* a tlačítko pro výstup z režimu celé obrazovky. Přetáčení podle slajdu v tomto režimu v základní podobě bude umožněno pouze na mobilních zařízeních s dotykovou

obrazovkou. Konkrétně v tomto režimu bude možno přejít na předchozí, resp. následující, slajd odpovídajícím jednoduchým pohybem prstu daným směrem na obrazovce.

Vzhled odlehčeného kontrolního panelu(*lite*) a *Layout* v režimu *portrait* jsou prezentovány na obrázcích 3.2 a 3.3.



Obrázek 3.2: Odlehčený (*lite*) kontrolní panel pro režim *fullscreen*

Obrázek 3.3: Layout přehrávače v režimu fullscreen portrét

3.4 Architektura

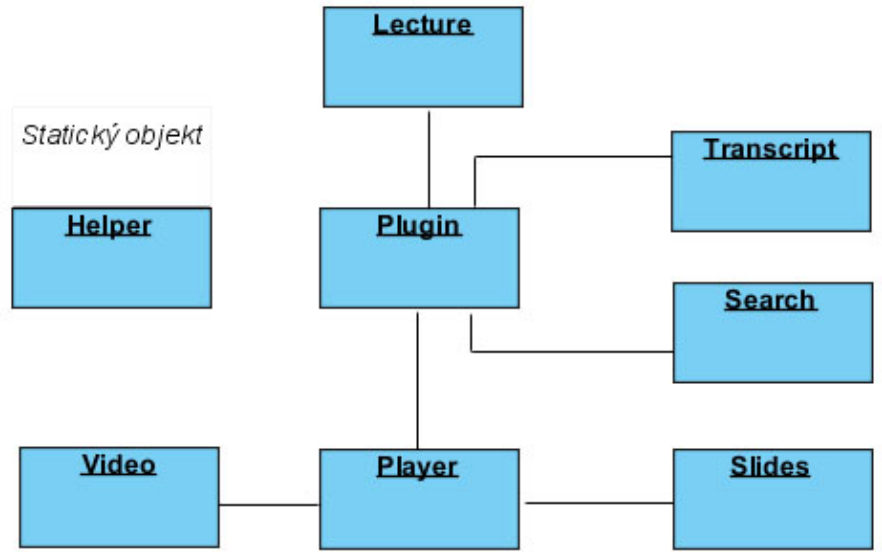
Z důvodu zvýšení robustnosti programu, usnadnění pochopení a tudíž i zjednodušení navázání dalšího vývoje, je nový přehrávač, přestože není cílový jazyk (JavaScript) třídně

orientovaným jazykem, navržen v souladu s objektově orientovaným paradigmatem a využívá hlavně konceptu zapouzdření.

Z hlediska architektury lze tedy veškerou funkčnost přehrávače rozdělit do několika objektů. Každý z objektů má přiřazenou určitou zodpovědnost a je určitým způsobem závislý na ostatních objektech.

Nový přehrávač Superlectures.com je organizován do sedmi objektů. Jedná se o objekty Plugin, Player, Video, Slides, Transcript, Search, Lecture a Helper.

Původním záměrem bylo vytvořit rovněž samostatný objekt pro každou záložku, což by znamenalo, že přibyly dva další objekty pro záložky "Stáhnout/Sdílet"a "Informace", nicméně po podrobnější analýze bylo zjištěno, že ani jedna z těchto dvou záložek, neobsahuje dostatek unikání funkcionality na to, aby jej bylo nutno obalit dedikovaným objektem. Veškerá funkčnost, týkající se těchto dvou záložek bude z tohoto důvodu obsažena v obecnějším, zastřešujícím objektu Plugin.



Obrázek 3.4: Objektový diagram přehrávače po inicializaci všech objektů

Plugin Z pohledu návrhových vzorů lze říct, že objekt Plugin vystupuje jako *Front-Controller* a *Creator* objektů ostatních.

Zodpovědností objektu Plugin je, v první řadě, sestavit kostru přehrávače, na nejobecnější úrovni, a následně vytvořit a předat řízení objektům následujícím, které přehrávač v dalších fázích postupně doplní o jimi spravovanou funkcionalitu.

Objekt Plugin, konkrétně, obsahuje řadu funkcí, které postupně sestavují výše zmíněnou kostru (šablonu) přehrávače, včetně šablon pro záložky "Stáhnout/Sdílet"a "Informace". Důležitou zodpovědností objektu Plugin je rovněž výběr jazykové mutace přehrávače a správa obecných modálních oken.

Po dokončení všech důležitých operací vytváří objekt Plugin instance objektů Player, Lecture, Transcript a Search, kterým předává kontrolu.

Player Zapouzdřuje atributy a funkce, které se týkají přehrávače slajdů a videa jako celku. Mezi takové funkce patří, ku příkladu, funkce odpovídající tlačítkům, která jsou situována

v pravé části kontrolního panelu přehrávače (obecná funkčnost přehrávače). Dále pomocné funkce pro dynamické překreslování tlačítek, jako reakce na interakce ze strany uživatele. A především pak funkce, které spravují přechody a celý pobyt přehrávače v režimu *fullscreen*. Objekt Player rovněž obsahuje klíčovou funkci, která dynamicky zajišťuje korektní zobrazení přehrávače vzhledem ke zbytku stránky a oknu prohlížeče po celou dobu běhu.

Objekt Player posléze vytváří instance objektů Videa a Slides, které jsou jeho potomky a doplňují přehrávač o další, konkrétnější, funkce.

Video Objekt Video je zodpovědný za videozáznam. Poskytuje funkce, které implementují funkce tlačítek kontrolního panelu videozáznamu a spravuje *progress-bar* videozáznamu. Jedná se o funkce prvků z levé části kontrolního panelu přehrávače jako *Play* a *Pause*, přehled o časové stopě videozáznamu, ovládání rychlosti přehrávání záznamu, ovládání hlasitosti a reakce na přetáčení pomocí *progress-baru* videa.

Objekt Video zasílá zprávy s požadavkem o synchronizaci objektům Slides a Transcript, pokud dojde k přetočení videa.

Slides Objekt Slides je zodpovědný za správu slajdů. Z hlediska architektury přehrávače je objekt Slides postaven na stejnou úroveň jako objekt Video. Objekt Slides ví, jaký slajd má v daný okamžik zobrazovat a má informace o tom, kdy zobrazit následující slajd. Rovněž zapouzdřuje funkce pro správu *progress-baru*, resp. přehledu, slajdů.

Objekt Slides zasílá zprávy s požadavkem o synchronizaci objektům Video a Transcript, pokud dojde ke změně slajdu.

Lecture Objekt Lecture uchovává data o aktuální přednášce. K těmto datům poskytuje celou sadu veřejných funkcí, skrze které umožňuje s daty pracovat ostatním objektům. Tato data mohou být buďto lehce předzpracovaná nebo neupravená a tudíž použita ve formátu, ve kterém je poskytuje API Superlectures.com.

Tato data dále využívají především objekty Video a Slides, které na základě informací poskytnutých skrze metody objektu Lecture prezentují jádro tohoto programu - video a slajdy.

Transcript Objekt Transcript řeší věškerou problematiku odpovídajícího záložce Transkript.

V první řadě doplní kostru přehrávače, sestavenou objektem Plugin o tuto záložku a následně provede definici veškeré její funkcionality.

Objekt Transcript zasílá zprávy s požadavkem o synchronizaci objektům Video a Slides, pokud dojde k přetočení ze strany výběru segmentu transkriptu.

Search Objekt Search, analogicky k objektu Transcript, řeší věškerou problematiku odpovídajícího záložce Vyhledávání.

Objekt Search zasílá zprávy s požadavkem o synchronizaci objektům Video, Slides a Transcript pokud dojde k přetočení ze strany výběru vyhledané fráze.

Helper Posledním objektem je objekt s názvem Helper. Helper je pomocný objekt, který obsahuje velmi obecné funkce, které svým typem nezapadají do žádného z předešlých objektů pluginu.

Z pravidla se jedná o funkce, které nemají nic přímo společného se sémantikou ostatních objektů nebo se jedná o funkce, které jsou využívány napříč několika objekty.

Jsou to funkce pro formátování času, URL adres, či například detekce orientace nebo dotykových zařízení.

Za běhu programu se nikdy nevytváří instance objektu Helper, proto lze tento objekt označit z hlediska návrhu za statický.

3.5 Technologie

Nová implementace přehrávače Superlectures.com využívá, z většinové části, stejné technologie jako implementace aktuální. Tím myšleno HTML, CSS, JavaScript, jQuery, Ajax, JWPlayer, XML a JSON.

Mimo tento vyčerpávající výčet technologií využívá nová implementace navíc knihovny pro jQuery jQueryUI a jQuery Mobile a šablonovací *engine* doT.js.

3.5.1 jQuery UI

JavsScriptová knihovna jQuery nabízí širokou škálu funkcí, které dokáží radikálně usnadnit, jinak mnohdy zdlouhavou, práci s DOMem. Existuje ale řada běžně používaných funkcí, které v knihovně jQuery najít nelze, jako jsou funkce pro práci s různými komponentami uživatelského rozhraní.

Tento fakt není nedostatkem jQuery, protože filozofie této knihovny je nabízet, co možná nejvíce odlehčenou, abstrakci nad JavaScriptem a prací s DOMem. Proto jakákoli taková rozšíření míjí zmíněnou filozofii a tudíž se do knihovny podobné nadstavby nepatří.

Tento "nedostatek"řeší jQuery UI neboli jQuery for User Interface [10].

JQuery UI je tedy množina funkcí pro zpracování uživatelských interakcí, efektů, *widgetů* a témat postavených přímo nad knihovnou jQuery.

Nabízí řadu rozhraní pro uživatelské interakce jako například Draggable a Droppable, které umožní pracovat s elementy DOMu ve stylu *Drag&Drop*, *widgety* pro populární prvky uživatelského rozhraní jako Datepicker, Progressbar, Tabs či Scroll a rozšiřující efekty typu Show, Hide či Easing [7].

V rámci tohoto přehrávače bude jQuery UI použito hlavně pro vytvoření speciálního posuvníku pro přehled slajdů.

3.5.2 jQuery Mobile

Další nadstavbou nad knihovnou jQuery je její rozšíření jQuery Mobile.

JQuery mobile je cíleno speciálně na responzivní webové stránky, kterým má za cíl pomoci snáze optimalizovat chod pro všechna mobilní zařízení.

Pro tento přehrávač je výhodné, že knihovna jQuery Mobile nabízí rovněž celou množinu abstrakcí nad dotykovými událostmi, jako jsou například komplexní události pro gesta typu `swipe`, `tap`, `taphold` nebo události nad virtualizovanou podobou myši `vclick`.

Celé zpracování dotykových událostí se může zdát jako banální záležitost, ale tato problematika skrývá obrovské množství skrytých problémů.

Podle standardu W3C mají moderní prohlížeče implementovat právě tři různá rozhraní, která souvisí se zpracováním dotykových událostí.

Prvním rozhraním je rozhraní Touch, které poskytuje několik atributů, jenž uchovávají informace o souřadnicích daného dotyku a to ve třech podobách. Relativně vůči *viewportu*

cílového zařízení a relativně vůči oknu prohlížeče v obou situacích, jak když je okno v základní pozici, bez jakéhokoli posuvu (*scroll*), tak relativně vůči oknu prohlížeče s tím, že uvažuje *offset* tvořený zmíněným posunutím. Posledním, velmi důležitým atributem je identifikátor DOM objektu, nad kterým byl dotek zaznamenán.

Rozhraním druhým je rozhraní TouchList, které se chová jako seznam všech provedených doteků, aby je bylo dále snazší interpretovat jako abstrakce událostí vyšší úrovně.

Třetím a posledním rozhraním je TouchEvent, které je nadstavbou nad zmíněným dvěmi předchozími rozhraními Touch a Touchlist. Rozhraní TouchEvent skládá jednotlivé dotyky a vytváří nad nimi abstrakce v podobě čtyř základních standardizovaných událostí. Tyto události jsou následující [13, 11].

- touchStart
- touchEnd
- touchMove
- touchCancel

Už od prvního pohledu je zřejmé, že tyto události jsou velmi nízkoúrovňové a ani se neblíží složitějším událostem nebo dokonce gestům, které se v dnešní době s oblibou používají.

Pro kvalitní implementaci plynulých a uživatelsky přívětivých gest je nutno uvažovat spoustu dobře nastavených časovačů, které dohromady tvoří ono plynulé gesto.

I v případě, kdy programátorovi povede nějaké jednoduché gesto kvalitně implementovat, je zde další problém v podobě různého chování, výše uvedených, standardních událostí v různých prohlížečích.

Problémem je, že mobilní prohlížeče, které podporují zpracování dotykových událostí musí rovněž zpracovávat události, které byly původně cíleny pouze na desktopová zařízení, jako je například událost `click`. Kdyby tomu tak nebylo, byly weby, které neimplementují podporu dotykových událostí naprosto nepoužitelné na zařízeních mobilních, což je samozřejmě extrémní a nežádoucí případ. Takže pro každou dotykovou událost na mobilním zařízení nejsou volány pouze standardní dotykové události, ale rovněž události souvisí s myší.

Aby řešení této situace ale nebylo tak přímočaré, jako ignorování dotykových událostí a prosté zpracovávání události `click`, existuje zde cca. 300ms prodleva mezi událostmi `touchStart` a `click`, aby bylo možno detekovat složitější gesta jako je například *scrolling* nebo aby zde jednoduše byla vestavěná prodleva pro případ, kdy si uživatel událost rozmyslí a chce ji zrušit.

Tento fakt by způsobit, že kdybychom se rozhodli ve skriptu reagovat pouze na událost `click` a ignorovat `touchStart`, aplikace by reagovala s touto 300ms odezvou a pocitově by působila "lenivě".

Kdybychom se naopak rozhodli, že skript bude reagovat na zachycení první události, tedy `touchStart`, je zde další problém. Tyto dvě události totiž nemusí vždy být spuštěny nad totožným elementem DOMu, takže je zde pravděpodobnost, že nad elementem, který sledujeme se událost `touchStart` nikdy ani neobjeví [4].

Mobilní prohlížeče pro zpracování těchto situaci používají řadu komplexních heuristik. Celý problém tedy vůbec není tak triviální, jak by se na první pohled mohl zdát. Proto je na místě svěřit zpracování dotykových událostí osvědčenému nástroji, jako je právě jQuery Mobile.

3.5.3 doT.js

Šablonovací engine doT.js je velmi malá, rychlá a úsporná JavsScriptová knihovna, která pro svůj běh nevyžaduje přítomnost žádných dalších závislostí. Tudíž je vhodným kandidátem pro přidání dalšího elementu požadované robustnosti do kódu přehrávače s velmi malou daní v rozdílu výpočetního času.

Celá podstata doT.js je, stejně jako u ostatních JavaScriptových šablonovacích systémů, založena na regulárních výrazech, pomocí kterých se v kódu šablony vyhledávají předdefinované značky s danou sémantikou a obsah těchto značek se překládá do anonymních JavaScriptových funkcí, které převádějí zpracování šablony na konkatenace HTML řetězců. Návratové hodnoty těchto anonymních funkcí jsou následně zkonkatenovány za sebe a výsledkem je požadovaný HTML kód.

Důvodem pro výběr právě doT.js byla právě rychlost vypozorovaná z provedeného benchmarku. Pro benchmark šablonovacích systémů byli vybráni tři populární kandidáti, jejichž výsledky byly porovnány s referenčním řešením, kterým je prosté uložení HTML kódu do skriptu v podobě řetězců.

Pomocí každého z kandidátů byla, v pěti měřeních, vygenerována kostra přehrávače. Těchto pět měření bylo, pro každý systém, provedeno odděleně na klasickém desktopovém zařízení a mobilním zařízení. Výsledky z jednotlivých měření byly zprůměrovány. Protože se průměrné hodnoty dob mezi desktopovými a mobilními zařízeními lišily velmi málo (v jednotkách milisekund), byly zprůměrovány i hodnoty napříč zařízeními.

Výsledky dopadly následovně.

- referenční způsob - ∼2ms x ∼3ms
- systém Underscore.js -∼ 96ms x ∼98ms
- systém HandleBars.js - ∼84ms x ∼92ms
- systém doT.js - ∼13ms x ∼16ms

Kapitola 4

Implementace nového přehrávače

Implementace vychází z fáze návrhu. Nový přehrávač je tedy implementován v jazyce JavaScript a je tvořen sedmi objekty, které jsou situovány v rámci jediného souboru. Tento soubor je strukturován jako plugin pro knihovnu jQuery.

V následujících sekcích bude diskutováno, proč byla zvolena právě struktura jQuery pluginu a jeh struktura. Dále budou postupně rozebrány a popsány všechny výše zmíněné objekty s důrazem na konkrétní implementační postupy a detaily.

4.1 jQuery Plugin

Při tvorbě rozsáhlejší aplikace je velmi důležité položit spolehlivý základ. Protože na tuto volbu přímo navazuje každý další krok vývoje aplikace, je nutno této volbě věnovat dostatečnou pozornost a zvolit tedy vhodnou strukturu.

K dnešnímu dni existuje rozvinutý ekosystém různých samostatných knihoven a modulů pro, jak JavaScript, tak samotnou knihovnu jQuery, které vystupují pod oficiálním názvem *jQuery Plugin*. Tyto pluginy nabízí vyčerpávající množství atraktivních funkcí a jsou tedy hojně využívány většinou moderních webů, které jsou na těchto technologiích vybudovány. Takové weby s výhodou využívají většinou jednotky až nízké desítky pluginů. Už kvůli tomuto faktu je vhodné aplikaci strukturovat tak, aby zbytečně nezatěžovala globální jmenný prostor a tudíž zamezila potenciálním možným kolizím jmenných prostorů s ostatními pluginy, jejichž autoři tuto skutečnost mohli opomenout.

V takové situaci může nastat stav, kdy jeden plugin přepíše, buď kompletně nebo částečně plugin jiný. Tato situace z pravidla působí cílovým uživatelům strastiplné chvíle při odhalování zmíněných konfliktů, které mohou nastat, v lepším případě, ihned ve fázi prvního spuštění aplikace s pluginem nebo dokonce, v horším případě, za běhu aplikace, případně se nečekaně mohou objevit po aktualizaci pluginu. A to vše jen díky nepozornosti autora.

Z výše zmíněných důvodů bude veškerá funkčnost přehrávače zapouzdřena do robustní struktury, *jQuery Plugin*. Tato struktura je, mimo jiné, velmi šetrná k jmennému prostoru aplikace, zabezpečuje korektní konkatenaci skriptu pluginu s jinými (i obfuskovanými) skripty a její další, nespornou, výhodou je, že umožní snadno navenek zpřístupnit pouze veřejné části (funkce) aplikace, zatímco privátní části od uživatele odstíní.

JQuery plugin je metoda, která využívá možnosti rozšířit (relace dědičnosti *extend*) prototyp jQuery objektu o dodatečnou funkcionalitu [6]. Tímto rozšířením je umožněno jQuery objektu využívat tuto funkcionalitu, pohodlně, kdekoli za běhu programu.

Neexistuje žádný oficiální, či pevně předepsaný způsob, jak strukturovat jQuery plugin.

Proto se za dobu existence jQuery objevila celá řada tzv. *boilerplates*. Termín *boilerplate* je možné volně popsat jako jakýsi standardní kus kódu, který se často opakuje a lze ho snadno upravit, na míru, pro potřeby konkrétní aplikace. Je tedy snadno znovupoužitelný. Z důvodu široké nabídky takovýchto znovupoužitelných kusů kódu nemá smysl znovu navrhovat již navržené a s výhodou využít situace.

Pro potřeby této aplikace je využit velmi populární *boilerplate* pod názvem jQuery Boilerplate, který je volně ke stažení a užívání pod MIT licencí.

Název samotného pluginy byl stanoven na `slplayer`, lze jej tedy jako plugin volat jednoduše `$('#element').splayer({})`.

Náhled na strukturu *boilerplate* viz. příloha B.

4.2 Plugin

Objekt Plugin je zastřešujícím objektem a spravuje nejobecnější chování pluginu přehrávače jako celku. Je tedy můstkem mezi knihovnou jQuery a objekty, které spravují konkrétní záležitosti týkající se přehrávaní záznamů.

4.2.1 Konfigurace

Prvním úkonem, který objekt Plugin v rámci konstruktoru provede je, že aplikuje uživatelem stanovou konfiguraci, čímž přepíše konfiguraci *defaultní*.

Konfigurace přehrávače se nachází v proměnné, která je globální v rámci všech objektů pluginu, s identifikátorem `defaults`.

Objekt `defaults` obsahuje řadu možností pro konfiguraci funkčnosti přehrávače, ale obsahuje jen a pouze takové atributy přehrávače, pro které má být umožněno, aby je uživatel cíleně mohl předefinovat - tedy veřejné. Veškeré privátní atributy, které jsou a mají být pro uživatele skryté, jsou zapouzdřeny v objektech, kterým jejich sémantika náleží.
Celý objekt `defaults` má následující podobu.

```
defaults = {
   template : 'basic/',
   requestFullscreenAllowed : true,
   swipeLeft : 'swipeleft',
   swipeRight : 'swiperight',
   lecture : {
      format : 'm4v',
      lectureID : '',
      lectureURL : ''
   },
   videoPlayer : {
      startTimeSec : 0,
      startTimeHms : "",
      autoStart : false,
      initialVolume : 50,
      aspectRatio : '4:3',
      primary : 'html5',
      controlPanelTimeHideUnder : 400,
   },
```

```
    slidesPlayer : {
        slideActualSize : 'src_large',
        slidePreviewSize : 'src_small'
    },
    transcript : {
        active : true
    },
    search : {
        active : true
    }
};
```

Všechny jednoduché atributy, které `defaults` obsahuje se týkají obecných vlastností přehrávače. V tomto případě jsou to pouze čtyři atributy, `template`, `requestFullScreenAllowed`, `swipeLeft` a `swipeRight`.

template Atribut `template` určuje adresář souborového systému, ve kterém jsou uloženy soubory definující šablonu přehrávače ve formátu .tpl.

Je možnost, že uživatel bude chtít předefinovat, ne vzhled přehrávače (z pohledu stylů CSS), ale HTML strukturu elementů přehrávače. V takovém případě pouze stačí vytvořit adresář s názvem "tématu", umístit jej na stejnou úroveň jako aktuálně používané téma *basic* a odpovídajícím způsobem upravit atribut `template`.

requestFullScreenAllowed Atribut `requestFullScreenAllowed` je booleovskká hodnota, která určuje, jestli bude v přehrávači povolen režim nativního *fullscreenu*.

swipeLeft a swipeRight Pár atributů `swipeLeft` a `swipeRight` určuje reakci na události typu *swipe* v režimu *fullscreen*, kde se tyto události používají pro změnu slajdu na předchozí/následující.

V kontextu těchto gest je jedna skupina uživatelů zvyklá provádět událost *swipeLeft* (gesto tažení prstem po obrazovce zařízení z prava do leva) s tím, že očekává že obsah se posune ve směru tahu prstu, což simuluje tažení celého obsahu gestem prstu.

Druhá skupina uživatelů je ale naopak zvyklá, že událost *swipeLeft* způsobí posun obsahu naopak směrem doprava, což odpovídá tahu jakéhosi plovoucího okénka nad statickým obsahem.

Jelikož pravděpodobně neexistují spolehlivé statistiky o rozdělení poměru těchto dvou skupin uživatelů, jsou tyto dva atributy součástí konfigurace s implicitním nastavením vyhovující první zmíněné skupině uživatelů.

Názvy atributů složených odpovídají názvům objektů přehrávače `Lecture`, `VideoPlayer`, `SlidesPlayer`, `Transcript` a `Search`. Vskutku, každý z těchto vnořených konfiguračních objektů ovlivňuje pouze vlastnosti svého jmenovce mezi objekty přehrávače. Toto rozdělení by mělo vnést do konfigurace větší přehlednost i v situaci, kdy počet konfiguračních atributů nabobtná do vysokých počtů.

Pro objekt `Lecture` lze konfigurovat následující vlastnosti.

format Určuje implicitní formát video složky záznamu, který bude použit pro přehrávání.

lectureID a lectureURL Jedinými povinnými atributy přehrávače je dvojice atributů `lectureURL` a `lectureID`.

`LectureURL` definuje cestu k dané události na serveru Superlectures.com a `lectureID` identifikuje konkrétní přednášku v rámci dané události. K jedné události z pravidla přísluší více, než jedna přednáška.

Tyto atributy, už z logiky věci, nemají žádnou definovanou implicitní hodnotu.

Konfigurovatelné vlastnosti objektu `videoPlayer` jsou následující.

startTimeSec a startTimeHms Atributy `startTimeSec` a `startTimeHms` reprezentují časové razítko, které určuje čas, ve kterém má videozáznam začít přehrávat. Tato vlastnost se využije, ku příkladu, k tomu, aby URL záznamu mohla být sdílena s ukazatelem na konkrétní čas.

autoStart Atribut `autoStart`, je booleovskou proměnnou, která rozhoduje, jestli se má záznam po načtení stránky, samovolně spustit.

Implicitní hodnotou je *false*, přehrávač tedy čeká na příkaz ze strany uživatele, ke spuštění záznamu. Nicméně aktuální přehrávač Superlectures možnost *autostartu* využívá, proto je tento atribut součástí konfigurace.

initialVolume Význam atributu `initialVolume` je zřejmý. Tento atribut určuje počáteční nastavení hlasitosti záznamu v procentech.

Defaultní hodnotou je 50%.

aspectRatio `AspectRatio` určuje poměr stran elementu obsahující videozáznam.

Defaultní hodnotou je 4:3, ale může být nastaven na hodnotu 16:9 v závislosti na tom, jaký je poměr stran přidružených slajdů, pro co možná nejlepší uživatelský zážitek.

`AspectRatio` je zároveň parametrem přehrávače JWPlayer.

primary Je parametr směřovaný čistě pluginu JWPlayer, který určuje, jestli má být přehrávače vyrenderován pomocí technologie HTML5 nebo pomocí technologie Flash.

Defaultní hodnotou je html5.

constrolPanelTimeHideUnder Atribut `constrolPanelTimeHideUnder` určuje čas v milisekundách, po jehož uplynutí se má v režimu *fullscreen* schovat kontrolní panel.

Defaultní hodnotou je 400 milisekund.

Konfigurační objekt pro `slidesPlayer` obsahuje dva atributy.

slideActualSize Určuje variantu zdrojového souboru slajdu, který se má použít pro zobrazení slajdu. Superlectures API nabízí celkem tři verze rozlišení slajdu `src_small`, `src_medium` a `src_large`.

Přehrávač defaultně načítá slajd v nejvyšším rozlišením.

slidePreviewSize Analogicky určuje variantu zdrojového souboru slajdu pro náhled v *progress-baru* slajdů. Defaultní hodnotou je, kvůli rychlosti prvotního načtení přehrávače a kvůli malým rozměrům náhledu, `src_small`.

4.2.2 Inicializace

V rámci běhu funkce `Plugin.init` se konečně dostává na řadu fáze vytváření dalších objektů, které se podílejí na běhu pluginu. Jedná se o objekty `Lecture`, `Transcript`, `Search` a `Player`.

Inicializační funkce objektu `Plugin` po předání řízení, jako první, vytváří objekt `Lecture`, který ze serveru Superlectures.com získá veškeré potřebné informace o dané přednášce. Následuje vytvoření objeku `Transcript`, který rovněž kontaktuje server Superlectures.com, odkud získá informace v podobě transkriptu záznamu. Po korektním získání transkriptu se provede vytvoření objektu `Search`, který pouze připraví adresu URL, pomocí které bude žádat server o zaslání výsledků vyhledávání. Jako poslední se po sléze vytváří objekt `Player`, který větví chod programu směrem k samotnému přehrávači.

Všem těmto objektům bude dále vyhrazena vlastní kapitola, proto jsou zatím zmíněny pouze z pohledu doby jejich vytvoření a inicializace.

4.2.3 Jazykové mutace

V okamžiku, kdy existuje objekt `Lecture`, lze jednoduchým dotazem na jím poskytované API zjistit, v jakém jazyce je daná přednáška vedena a nastává vhodná chvíle pro volbu jazykové mutace přehrávače.

Přehrávač nabízí, po vzoru, aktuálního přehrávače Superlectures.com dvě jazykové mutace a to českou a anglickou. Do obou jazyků jsou tedy přeloženy všechny popisky, fráze, či věty, které se nacházejí ve všech štítcích přehrávače.

Všechny tyto fráze jsou situovány v jediné funkci objektu Plugin pod identifikátorem `Plugin.setLanguage`, která obsahuje dva vnořené objekty s názvy `cs` a `en`, reprezentující jazyky Čeština a Angličtina.

Rozhodnutí o výsledné mutaci se provádí následující podmínkou.

```
if(this.lecture.getAttributes().language == 'cs')
   $.extend(languageString, cs):
else
   $.extend(languageString, en);
```

Výsledkem je patřičné rozšíření globálního objekt `languageString` daným objektem jazykové mutace, čímž je jazyk přehrávače kompletně vyřešen a není třeba nic dalšího nastavovat.

4.2.4 Sestavení HTML kódu

Po inicializaci objektu se dostává ke slovu nejrozsáhlejší část kódu objektu `Plugin`, která se stará o vygenerování základní (statické) struktury HTML kódu přehrávače.

Celý proces generování je spuštěn funkcí objektu `Plugin` - `Plugin.build`.

V tomto bodě je na místě zmínit, že v rámci kódu přehrávače je dodržována konvence, napříč všemi objekty, taková, že pokud daný objekt za běhu generuje nějaký dynamický obsah, je proces generování vždy zapouzdřen do funkce s názvem `Objekt.build`. Případně, pokud generovaný úsek kódu tvoří více samostatných logických celků, je generování rozděleno do několika funkcí, které zpracovávají odpovídající logické celky. Identifikátor každé takové funkce má předponu *build* a jeho přípona dále specifikuje logický celek, který generuje.

Tyto logické celky jsou následně spojeny ve zmíněné, zastřešující, funkci `build`.

Funkce build

Jak bylo konstatováno v technologiích, používá tento přehrávač pro generování HTML kódu šablonovací systém doT.js.

DoT.js tedy očekává na vstupu šablonu a aplikační data. V rámci procesu kompilace tyto dvě entity patřičně propojí a vrátí standardní HTML kód.

V tomto bodě je nutno provést další rozhodnutí. Jedná se o rozhodnutí, jak nejlépe rozdělit logiku zpracování vytváření šablony mezi dvě entity, kterými jsou *Frontend controller* (objekt `Plugin`) a samotnou šablonu, kde je možno využít značek, které nabízí *framework* doT.js. Přesněji řečeno, jaké množství logiky je vhodné přenechat právě šabloně, aby šablona nebyla zahlcena výpočetní logikou, což by mělo za následek zpomalení její kompilace a zároveň by došlo k porušení logiky konceptu MVC.

Zvolená implementace v rámci *controlleru* provádí patřičné předzpracování dat, které pak předává šabloně tak, že se v rámci logiky šablony řeší jen a pouze situace podmíněného vypsání daného kusu kódu, případně situace, kdy je nutno přes cílový atribut iterovat.

V rámci přílohy C je uvedeno několik konkrétních příkladů práce s doT.js v šabloně přehrávače

Za běhu funkce `Plugin.build` jsou aplikační data uložena v rámci objektu `context` s následující strukturou.

```
var context = {
   tabs : self.buildTabs(),
   info : self.buildInfo(),
   download : self.buildDownload(),
   share : self.buildShare(),
   transcript : self.transcript.transcript,
   search : self.buildSearch(),
   additional : self.buildAdditional()
}
```

Každý z atributů objektu `context` odpovídá jednomu logickému celku struktury HTML kódu a je tedy patřičně naplněn odpovídající funkcí s předponou *build*.

buildTabs

Se stará o vygenerování základní struktury všech štítků v horní části přehrávače.

buildInfo

Má na starost vygenerování obsahu štítku Informace.

V rámci funkce `buildInfo` se používá několik pomocných funkcí, které pomáhají vhodně předzpracovat obsah této záložky, aby se odlehčila zátěž při kompilaci šablony.

buildDownload

Řeší sestavení částizáložky Stáhnout pro záložku Stáhnout/Sdílet.

buildShare

Analogicky, řeší sestavení části Sdílet pro záložku Stáhnout/Sdílet.

buildTranscript a buildSearch

Tyto dvě funkce mají na starost sestavení kostry záložek Transcript a Search. Daná funkcionality se následně generuje až v rámci odpovídajících objektů `Transcript` a `Search`.

buildAdditional

Obsahuje několik dalších různých atributů pro korektní sestavení šablony přehrávače.

buildEvents

Poslední funkcí objektu `Plugin` je funkce, které se stará o registraci všechno událostí, kterým tento objekt naslouchá.

Jedná se o veškeré obecné uživatelské interakce v rámci záložek, kterou nejsou záložkami Video, Transkript a Search, včetně základní logiky přepínání záložek. Zpracování událostí pro hlavní záložku Video je rozloženo do objektů `Player`, `videoPlayer` a `SlidesPlayer`.

4.2.5 Modální elementy

Přehrávač používá na několika místech funkcionalitu tzv. *modálních elementů*. Logiku modálních elementů využívá několik objektů přehrávače, nicméně z úsporných důvodu a koheze bude veškeré modální chování zmíněno v této sekci.

Za *modální element* se ve webovém prostředí označuje prvek stránky, který dočasně překrývá určitou část stránky tak, že není uživatelovi umožněno interagovat s elementy (vrstvami) pod tímto elementem. Toto chování je velmi podobné chování modálních oken v prostředí operačních systémů.

Tento princip je vhodný např. pro implementaci funkcionality náhledu slajdu, při prohlížení slajdů, náhledu na detail vyhledávané fráze v audiu a dokonce i režimu celé obrazovky. Funkcionalita záložek rovněž odpovídá logice modálního chování.

Modální prvek je v přehrávači zastoupen prvkem `modal-content` a zmíněné chování je vyřešeno pomocí odpovídajících CSS vlastností v jeho definici.

```
#modal-content {
    overflow: auto;
    position: absolute;
    height: 100%;
    width: 100%;
    z-index: 4;
}
```

Pomocí nastavení position na absolute, width a height na 100% a z-indexu na hodnotu vyšší, než hodnoty z-indexů zbytku přehrávače lze toto chování snadno vynutit.

4.2.6 Předání control-flow

V momentě, kdy objekt `Plugin` připraví celou kostru přehrávače, nezbývá mu nic, než konečně předat *control-flow* následníkovi `Player`.

4.3 Player

V tomto okamžiku je připravena kostra přehrávače, dochází k doplnění funkcionality samotného přehrávání videa a slajdů.

4.3.1 Detekce slajdů

Inicializační funkce objektu `Player`, v první řadě, provede detekci přítomnosti slajdů u záznamu. To vše, voláním odpovídající funkce `Lecture.slidesPresent`, objektu `Lecture`. Pokud jsou slajdy u záznamu přítomny, vytvoří instance objektů `slidesPlayer` a `videoPlayer` a následně patřičně upraví nastavení *layoutu* přehrávače tak, aby byla levá polovina vyhrazena přehrávači videa a polovina pravá přehrávači slajdů.

Samozřejmě, pokud funkce `Lecture.slidesPresent` odpoví záporně, objekt `slidesPlayer` není třeba vytvářet a *layout* se přizpůsobí pouze videu.

4.3.2 Fullscreen

Veškeré zpracování, jak přechodu do režimu *fullscreen*, tak návratu, je implementováno v rámci funkce `Player.fullscreen`.

Funkce `Player.fullscreen` poskytuje dvě různé implementace režimu celé obrazovky.

Základním režimem je režim, kdy je celou obrazovkou myšlen objekt DOMu `Window`. Jinými slovy přehrávač v tomto režimu zabere veškeré volné místo v rámci okna prohlížeče. Tato implementace používá základních, standardizovaných CSS vlastností a tudíž je z hlediska podpory bezpečná a je podporována na všech dostupných platformách. Pro přechod do tohoto režimu stačí elementu přehrávače `#player` přiřadit připravenou CSS třídu, která vyjme element z toku dokumentu a roztáhne jej tak, aby zabíral celý *viewport* prohlížeče. Tohoto chování lze docílit například následující definicí třídy `fullscreen`, která je velmi podobná definici funkcionality modálního okna, s výjimkou vlastnosti `position`, která je zde nastavena na `fixed`.

```
.fullscreen {
    position: fixed;
    height: 100%;
    width: 100%;
    left: 0;
    top: 0;
    z-index: 10;
}
```

Přehrávač obsahuje ale i rozšířenou implementaci režimu celé obrazovky, využívající JavaScriptového API *requestFullscreen*. API *requestFullscreen* dovoluje jednoduše danému elementu a jeho potomkům zabrat veškeré volné místo na uživatelově obrazovce tak, že dočasně eliminuje veškeré rozhraní prohlížeče a ostatních aplikací, čímž docílí efektního zobrazení.

Veškeré zpracování přechodu do režimu celé obrazovky, a naopak, je zapouzdřeno ve funkci `Player.fullScreenToggle`, která je obálkou nad voláními funkci API *requestFullscreen*. `FullScreenToggle` je ve vhodný moment volána zmíněnou funkcí `Player.fullscreen`.

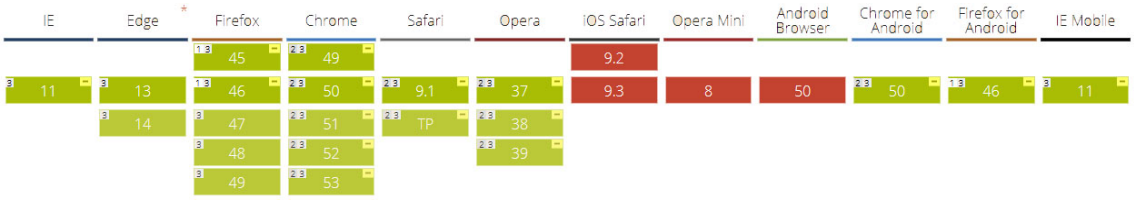
Toto API je stále v experimentální fázi vývoje a není tedy standardizováno a rovněž není, ani z daleka, podporováno na všech dostupných platformách. Nicméně, co se týče

platforem desktopových, je API podporováno ve všech prohlížečích, kromě mobilní verze iOS Safari. Tudíž vzhledem k velmi efektní a uživatelsky atraktivní funkcionalitě stojí za využití. Z důvodu neúplné podpory je nutno při volání metod API brát na zřetel, jaký prohlížeč uživatel používá a rozlišit různé implementace daných metod API s využitím standardních Vendor prefixů.

Z důvodu použití takto, zatím, nestandardní technologie, bude implementace nativního *fullscreen* diskutována detailněji.

Na následujícím obrázku 4.1 je uvedena podpora *requestFullScreen* API v prohlížečích, které mají k 5.5.2016 podíl použití mezi uživateli alespoň 1%.

Zelenými odstíny barev jsou označeny prohlížeče podporující toto API a červenými naopak prohlížeče nepodporující *requestFullScreen* API. U každého uvažovaného prohlížeče je zároveň uvedena jeho verze.



Obrázek 4.1: Tabulka kompatibility prohlížečů vůči API requestFullScreen

Z důvodu statusu podpory lze v přehrávači užívání *requestFullscreen* API úplně vypnout již při volání metody `Player.fullscreen(False)`. Nicméně i pokud je použití API povoleno voláním `Player.fullscreen(True)`, provádí se před voláním metody API kontrola kompatibility pro daný prohlížeč, který uživatel momentálně používá. Pro tento případ nabízí API čtveřici funkcí.

```
document.body.requestFullscreen
document.body.msRequestFullscreen
document.body.mozRequestFullScreen
document.body.webkitRequestFullscreen
```

Dané varianty jsou odděleny pro prohlížeče rodin -o-, -ms-, -moz-, -webkit-. Tyto funkce vrací *True*, pokud daný prohlížeč API podporuje a *False* v opačném případě. Pokud je API podporováno, stačí pak pro přechod do režimu celé obrazovky (resp. návrat z tohoto režimu) volat odpovídající z metod.

```
document.body.requestFullscreen() //document.exitFullscreen()
document.body.msRequestFullscreen() //document.msExitFullscreen()
document.body.mozRequestFullScreen() //document.mozCancelFullScreen()
document.body.webkitRequestFullscreen() //document.webkitCancelFullScreen()
```

Metodu lze implementoval jako *toggle* (překlápění) pomocí následujících složené podmínky, která využívá poslední z funkcí, které API *requestFullscreen* nabízí. Tato rodina metod vrací *True*, pokud se nějaký element právě nachází v režimu *fullscreen* a *False* v případě opačném. Složením tedy dostaneme výraz, který lze kladně vyhodnotit ve všech prohlížečích, podporujících API.

```
if (!document.fullscreenElement &&
```

```
    !document.msFullscreenElement &&
    !document.mozFullScreenElement &&
    !document.webkitFullscreenElement)
```

I tento režim používá CSS třídu `.fullscreen`.

Poslední záležitostí, kterou je nutno při používání API zohlednit je fakt, že při vstupu do režimu *fullscreen* nabídne prohlížeč uživateli možnost výstupu z tohoto módu pomocí klávesy ESCAPE. Tuto funkčnost není možno nijak vypnout nebo filtrovat, jelikož se jedná o ochranný prvek, který chrání uživatele před potenciálním uváznutím v tomto režimu, kterého by bylo možno zneužít. Pokud tedy uživatel ve *fullscreen* módu a stiskne klávesu ESCAPE, dojde k okamžitému volání funkce API *exitFullscreen* (resp. jedné z funkcí pro ukončení režimu) a tedy nedojde k ostatním krokům implementovaným ve funkci `Player.fullscreen`, které jsou nutné pro korektní návrat do normálního režimu a zachování konzistentního stavu přehrávače.

Tuto situaci lze řešit následující konstrukcí.

```
var fsChangeFilter = false;

$(window).on('webkitfullscreenchange mozfullscreenchange
              fullscreenchange MSFullscreenChange',
   function() {
      fsChangeFilter = !fsChangeFilter;

      /* True == requestFullscreen enabled */
      if(self.isFullscreen && !fsChangeFilter)
         self.fullscreen(self.config.requestFullscreenAllowed);
   }
);
```

Výše uvedená konstrukce "odchytává"události *webkitfullscreenchange*, *mozfullscreenchange*, *fullscreenchange* a *MSFullscreenChange*, které nastávají v případě přechodu do a z režimu *fullscreen* a to i tehdy, pokud je použito ukončení pomocí klávesy ESCAPE. Pomocí překlápění proměnné `fsChangeFilter` lze vyfiltrovat každý přechod do režimu *fullscreen* a reagovat pouze na přechody z řežimu *fullscreen*. Výsledkem této konstrukce je volání metody `Player.fullscreen` v případě, že uživatel stiskne ESCAPE, tudíž i v tomto okrajovém případě dojde ke korektnímu přechod do normálního režimu přehrávače.

4.3.3 Zarovnání videa a slajdů

Nejrozsáhlejší funkcí objektu Player je funkce `Player.update`, která je volána při každé změně velikosti okna prohlížeče a při změně orientace zařízení.

Funkce zajišťuje dynamické responzivní chování přehrávače nejen v módu celé obrazovky, ale rovněž v situacích, kdy uživatel manipuluje s oknem prohlížeče. Jedná se o část responzivního chování, které není možné zajistit pouze za použití CSS z důvodu, že rodičovské elementy element přehrávače videa a slajdů pracují s procentuálně definovanými rozměry - viz. dále.

Řízení funkce je větveno dvěmi směry, které pokrývají situace kdy je přítomen přehrávač slajdů a kdy naopak není. Pro možnost, kdy se jedná pouze o přehrávač videozáznamu lze

spoléhat na vestavěnou reposnzivitu pluginu JWPlayer a tudíž celého synovského objektu videoPlayer. V situaci, kdy slajdy přítomny je nutno provádět dodatečnou režii a to v obou módech orientace - *landscape* a *portrait*.

Pricip řešení responzivity přehrávače spočívá v nastavení výšky rodičovským elementům přehrávače videa a přehrávače slajdů, na pevnou hodnotu a nastavení šířky na 100% šířky celého přehrávače.

Tímto nastavením se zaručí dokonalé vyplnění volného místa těmito dvěmi *wrappery*. Samotný výpočet výšky je triviální.

V dalším kroku dojde nastavení velikostí samotných prvků videa a slajdů. Nyní je ale nutno rozlišit orientace *landscape* a *portrait*. K detekci orientace je použita jednoduchá funkce objektu `Helper` - `Helper.getOrientation`. Vstupní podmínkou je fakt, že JWPlayer v responzivním módu ignoruje jakékoli nastavení výšky. Je proto nutno provádět nastavení skrze šířku elementů a znalost poměru stran.

Výpočet šířky v módu *landscape* se dá, bez ohledu na optimalizaci, reprezentovat následující podmínkou. Vstupní hodnota `innerHeight` reprezentuje šířku *viewportu* prohlížeče a `height` je výška vrácena funkcí `Player.getUsableHeight`.

```
var aspectWidth = height/3*4,
    viewportWidth = $(window).innerWidth()/2,
    width = 0;

if(viewportWidth < aspectWidth)
   width = viewportWidth;
else
   width = aspectWidth;
```

Tato konstrukce zaručí že se jednak zachová poměr velikostí hran elementu, který je v této situaci nastaven na 3:4 a zároveň elementy nebudou přesahovat za hranice kontejneru. Analogicky by bylo možné popsat výpočet šířky pro element při orientaci *portrait*.

Poté, co jsou vypočteny velikosti všech potřebných elementů, dojde k zarovnání na střed, pomocí nastavení vhodných vnitřních okrajů (*padding*) *wrapperům*.

4.3.4 Detekce aktivity uživatele

Poslední podstatnou záležitostí, kterou objekt `Player` řeší, že je detekce aktivity uživatele.

Informace o aktivitě, resp. neaktivitě, uživatele se následně využívá v režimu *fullscreen*, kde se při změně stavu uživatele z *active* na *inactive* schovává odlehčený kontrolní panel.

Aby nedocházelo v k zahlcení přehrávače zpracováním všech událostí, které signalizují aktivitu uživatele (každá akce obvykle vyvolá funkci *handleru* události), nastavuje každý událost pouze stavovou proměnnou, kterou pak stačí ve vhodně vzorkovat. Tím lze dosáhnout principiálně stejného výsledku, nicméně s ušetřením i znatelného množství výkonu.

S touto proměnnou následně pracuje handle funkce `Player.activityCheck`, která ji každé 0.25 vteřiny vzorkuje, a v případě, že je uživatel momentálně aktivní, nastaví aktivitu na *True* a zeptá se funkce `Player.userActive`, jestli je kontrolní panel fyzicky (z hlediska CSS) aktivní nebo schován.

Pokud je už schován, tak jej zpět zobrazí. Dále pak znovu nastaví 2-vteřinový odpočet neaktivity. Pokud tento odpočet doběhne do konce, je ovládací panel schován již zmíněnou funkcí `Player.userActive`.

```
activityCheck : function() {
   var self = this;

    if (this.userIsActive) {
      this.userIsActive = false;

      if(!this.userActive())
         this.userActive(true);

      clearTimeout(this.inactivityTimeout);

      this.inactivityTimeout = setTimeout(function() {
         if (!self.userIsActive)
            self.userActive(false);
      }, 2000);
   }
}
```

Způsobů, jak animovaně schovat prvek stránky je více. Je možno se na problém podívat ze strany JavaScriptu, resp. knihovny jQuery, kde je k dispozici rodina vestavěných funkcí z kategorie *jQuery Effects* jako jsou *animate*, *fade*, *show*, *hide*, *toggle* apod. nebo ze strany CSS3, kde se nabízí vlastnosti *animate* a *transition*.

Pro tento případ byla zvolena cesta, následující CSS3 přechodu.

```
#control-panel-lite.userInactive {
   display: block;
   visibility: hidden;
   opacity: 0;
   -webkit-transition: visibility 1.5s, opacity 1.5s;
   -moz-transition: visibility 1.5s, opacity 1.5s;
   -ms-transition: visibility 1.5s, opacity 1.5s;
   -o-transition: visibility 1.5s, opacity 1.5s;
   transition: visibility 1.5s, opacity 1.5s;
}
```

Z důvodu plynulé reakce reaguje funkce na mobilních zařízeních na událost *touchStart* místo události *tap*.

4.4 VideoPlayer

Jelikož API JWPlayeru nenabízí možnost doimplementovat vlastní tlačítka do kontrolního panelu přehrávače, resp. s kontrolním panelem vůbec nějak manipulovat, zapouzdřuje objekt `videoPlayer` kromě vytvoření tzv. *chromeless* instance JWPlayeru, což je JWPlayer bez kontrolního panelu, rovněž implementaci tlačítek kontrolního panelu.

Objekt `videoPlayer` tedy figuruje jako nadstavba JWPlayeru, která doimplementovává veškeré jeho ovládací prvky pro potřeby tohoto přehrávače.

Většina funkcí tohoto objektu jsou funkce velmi nízkoúrovňové a neobsahují tudíž mnoho komplikovanějších implementačních detailů, které by bylo nutno hlouběji analyzovat, či diskutovat, a zároveň většina těchto funkcí obsahuje mnoho společných rysů.

Pro příklad - implementace funkcionality ovládání hlasitosti je z pohledu práce s DOMem, založena na stejných principech jako implementace *progress-baru* videozáznamu.

4.4.1 Inicializace

V rámci inicializační funkce objekt `videoPlayer` provádí dvě hlavní činnosti.

První činností je registrace veškerých událostí, spojených s uživatelskou interakcí s kontrolním panelem videopřehrávače a registrace intervalů, které jsou nadále použity pro aktualizaci periodicky se měnících částí přehrávače, jako je například informace o uplynulém času záznamu, či periodické překreslování *progress-baru* videopřehrávače.

Druhou činností je pak vytvoří samotné instance *chromeless* verze JWPlayeru s následujícími parametry.

```
this.playerInstance.setup({
   primary : self.config.primary,
   file : self.player.lecture.getVideo(),
   image : self.player.lecture.getThumbnail(),
   aspectratio : self.config.aspectRatio,
   width: playerWidth,
   controls : false,
   autostart : self.config.autoStart
});
```

Veškeré informace potřebné pro konstrukci instance JWPlayeru jsou buďto získány z konfiguračního objektu nebo z objektu `Lecture`.

Atribut controls zajistí, že instance bude vytvořena bez kontrolního panelu.

4.4.2 Princip synchronizace

Z pohledu logiky veškerého přetáčení v rámci celého pluginu figuruje objekt `VideoPlayer` jako referenční bod, vzhledem ke kterému se synchronizují ostatní prvky přehrávače - jmenovitě jde o objekty `SlidesPlayer` a `Transcript`.

Toto se děje z důvodu, že jen a pouze objekt `VideoPlayer` má přístup k referenčním událostem, které generuje samotný JWPlayer. Událostí, která se používá pro synchronizaci celého přehrávače je událost *onTime*, kterou JWPlayer uvolňuje periodicky, každých 100 milisekund.

Tato událost je sice nahraditelná generickým intervalem, definovaným pomocí JavsScriptu, nicméně pokud přehrávač naslouchá přímo události JWPlayeru, bude vždy vykazovat plynulejší chování, než kdyby naslouchal obecné události, která bude vůči události *onTime* nějakým způsobem posunutá. Další nespornou výhodou je rovněž fakt, že JWPlayer událost *onTime* vysílá jen tehdy, pokud je videozáznam opravdu přehráván a tedy zbytečně neplýtvá výkonem v případě, že přehrávač nepřehrává. Při využití generického intervalu by bylo nutno tuto logiku zbytečně doimplementovávat.

Objekt `VideoPlayer` tedy této události, s výhodou, naslouchá a synchronizuje podle ní veškeré vnitřní záležitosti.

Objekty `SlideSPlayer` a `Transcript` se pak periodicky dotazují (*polling*) objektu `VideoPlayer` na aktuální čas, čímž se vzhledem k tomuto referenčnímu objektu automaticky synchronizují.

Mimo automatickou synchronizaci je nutno tyto tři objekty navzájem synchronizovat při přetáčení *on-demand*, tzn. přetáčení, které je vyvoláno některou z uživatelských akcí jako třeba kliknutí na *progress-bar* videa, kliknutí na *progress-bar* slajdů, kliknutí na některý ze segmentů transkriptu nebo vyhledanou frázi.

Pro tento případ obsahuje každý, z těchto čtyřech, objektů funkci pro dynamickou synchronizaci pod identifikátorem `syncToRewind` a konvenční sadu funkcí pro různé způsoby přetáčení, které vždy začínají předponou *rewind* a jejich přípona pak určuje konkrétní způsob přetáčení v rámci daného objektu. Tyto funkce jsou pak volány objektem, který provedl přetočení a zasílá tímto způsobem ostatním objektům příkaz k synchronizaci.

Objekt `VideoPlayer` tuto funkci obsahuje pouze jednu a to pod identifikátorem `VideoPlayer.rewindToTime`, jenž zpracovává přetočení videozáznamu kliknutím na jeden ze dvou *progress-barů* přehrávače.

Funkce rewindToTime

Funkce `VideoPlayer.rewindToTime` je nízkoúrovňová funkce, která přepočítává souřadnice pozice uživatelovy interakce s *progress-barem* videozáznamu, na odpovídající čas videozáznamu pro *on-demand* přetočení.

Na základě typu *progress-baru* funkce získá DOM element odpovídajícího *progress-baru* a následně konvertuje souřadnice výskytu oné interakce na čas záznamu a provede přetočení.

Pro hladší chod, aby objekty `SlidesPlayer` a `Transcript` nemusely čekat na sepnutí automatické synchronizace, rovněž funkce provádí okamžitou synchronizaci následujícími podmíněnými voláními.

```
/* Change slide (if there are slides) */
if(this.player.slidesPlayer)
   this.player.slidesPlayer.syncToRewind(time);

/* Same for transcript */
if(this.player.transcript)
   this.player.transcript.syncToRewind(time);
```

Funkce syncToRewind

Analogicky k funkcím `syncToRewind` objektů `SlidesPlayer`, `Transcript` a `Search` poskytuje funkci pro okamžitou synchronizaci také objekt `VideoPlayer`.

Funkce má následující podobu, kde první podmínka řeší situaci, ve které je přehrávač zastaven nebo pozastaven, ale uživatel vyžádá přetočení. V tomto případě tedy přehrávač prvně simuluje uživatelův klik na tlačítko *Play*. Poté už jednoduše, pomocí volání API JWPlayeru, provede přetočení záznamu a docílí synchronizace s ostatními objekty.

```
if(this.playerInstance.getState() != 'playing')
   $('#play').trigger('click');

this.playerInstance.seek(time);
```

4.5 SlidesPlayer

Párovým objektem k objektu `VideoPlayer` je `SlidesPlayer`, který se stará o veškeré záležitosti související se synchronním přehráváním slajdů.

4.5.1 Inicializace

Ihned po získání řízení objekt `SlidesPlayer` získává skrze rozhraní objektu `Lecture` všechny dostupné slajdy pro daný záznam. Tento objekt uloží do vnitřního atributu `SlidesPlayer.slides` a s tímto objektem nadále pracuje jako s jednosměrně vázaným seznamem tak, že si atributu `this.nextSlide` udržuje informaci o času nástupu dalšího slajdu, relativně vůči slajdu aktuálnímu. A v atributu `this.lastSlide` urdžuje index poslední slajdu seznamu, který funguje jako zarážka.

4.5.2 Sestavení přehledu

Progressbar slajdů je sestaven dynamicky z důvodu, že v době injekce statické kostry pluginu do HTML kódu stránky ještě není dokončeno zpracování asynchronního dotazu, který vrací informace o dané přednášce a tudíž tedy není znám ani počet slajdů pro vykreslení.

Celý proces generování řeší dvě další funkce, z rodiny funkcí s prefixem *build*, konkrétně `SlidesPlayer.buildSlidesBar` a `SlidesPlayer.buildSlider`.

Přehled slajdů je sestaven jako jako statický element (není responzivní), jelikož při vytváření HTML elementu *img* daného slajdu je sice obrázek součástí DOMu, ale není vykreslen. Proto je nutno nastavit rozměry na statické hodnoty, které se předpočítávají ze znalosti poměru stran slajdu. Toto se děje z důvodu, že se obrázky fyzicky načítají až po načtení kompletního DOMu stránky, zatímco plugin se začíná vkládat do kódu stránky již při události *document.ready*, která je spuštěna právě po načtení DOMu. A tedy dříve, než mohou být načteny všechny *img* elementy. Tento fakt by znemožnil konzistentní generování speciálního elementu *slideru* pro *progress-bar* slajdů.

Z pohledu CSS je samotný *progress-bar* slajdů řešen typicky následovně.

```
#slides-bar {
    height: 100%;
    background-color: black;
    overflow-x: auto;
    overflow-y: hidden;
    white-space: nowrap;
}
```

Kombinace vlastností *overflow-x: auto* a *white-space: nowrap* vynutí požadováné chování, kdy je *progress-bar* tak široký, aby dokázal obalit všechny slajdy a zároveň v ose x není zalamován, ale zůstává na jediném "řádku".

Konkrétní slajdy pak jsou plovoucími elementy.

4.5.3 Detekce dotykové obrazovky

Přehrávač poskytuje možnost nahrazení generického HTML posuvníku atraktivnější verzí. Toto nahrazení je nicméně prováděno pouze na zařízeních bez dotykové obrazovky.

Důvodem je fakt, že samotné HTML poskytuje výborný posuvník pro dotyková zařízení, který reaguje na dotykové události a je výborně optimalizovaný.

Nicméně proces detekce podpory dotykových událostí na cílovém zařízení je velký problém. Dokonce tak velký, že zmíněná detekce ze strany prohlížeče aktuálně není nijak spolehlivě implementována.

Na samotný prohlížeč se lze podívat jako na velmi omezený *sandbox*. Kód běžícího skriptu se může skrze prohlížeč dostat k řadě informací ve formě HTML, CSS a JavaScriptového API, které samotný prohlížeč skriptům nabízí a tudíž rozsah těchto API striktně definuje k jakým informacím má daný skript přístup.

Pokud tedy je požadavek na zjištění, jestli cílový systém podporuje danou funkci, může se skript tázat, jestli pro ni prohlížeč poskytuje dané API nebo experimentálně sledovat, jak se skript za běhu v dané situaci zachová a pomocí této informace odvodit, jestli je funkce přítomná, či není.

Existují různé přístupy, jak těmito způsoby implementovat "detekci"dotykové obrazovky, mezi něž patří třeba rozsáhlejší knihovna *Modernizr* pro JavaScript. V rámci tohoto skriptu byl ale využit jiný přístup skrze JavaScript, který je definován v následující funkcí objektu `Helper.isTouchDevice`, jíž využívá i objekt `SlidesPlayer`.

```
isTouchDevice : function() {
   return typeof window.ontouchstart !== 'undefined';
}
```

Princip je jednoduchý, funkce se ptá JavaScriptového API prohlžeče, jestli je v něm definována standardní událost *ontouchstart* a předpokládá, že pokud tato událost definována je, pak se jedná o dotykové zařízení.

Funkce `Helper.isTouchDevice` se ukázala být konzistentní v rámci testování (viz. kapitola testování), kdy se chovala korektně a neprodukovala žádné *False positive* výsledky.

Funkce buildSlider

Sestavení samotného posuvníku se děje v rámci zmíněné funkce `SlidesPlayer.buildSlider`, které je podmíněně volána po vytvoření *progress-baru* slajdů za podmínky podpory dotykových událostí.

Při splnění této podmínky se elementu *progress-baru* slajdů vypne HTML posuvník, pomocí CSS vlastnosti *overflow-x: hidden*.

Implementace posuvníku je postavena na *Slider Widgetu*, který poskytuje knihovna jQuery UI.

4.5.4 Centrování aktivního slajdu

Po určité době neaktivity dojde k automatickému vycentrování *progress-baru* slajdů k aktuálně přednášenému slajdu. Toto chování lze implementovat pomocí nastavení *offsetu* slajdů vůči jejich kontejneru.

Centrování je prováděno v pravidelném intervalu. Tento interval je nicméně nutno vynulovat a znovu nastavit při jakékoli interakci uživatele s *progress-barem* slajdů, aby nedocházelo k situacím, kdy uživatel prochází slajdy, které se nechtěně vycentrují na aktuální slajd. A to v obou případech, jak pro klasická zařízení, tak pro zařízení s dotykovou obrazovkou.

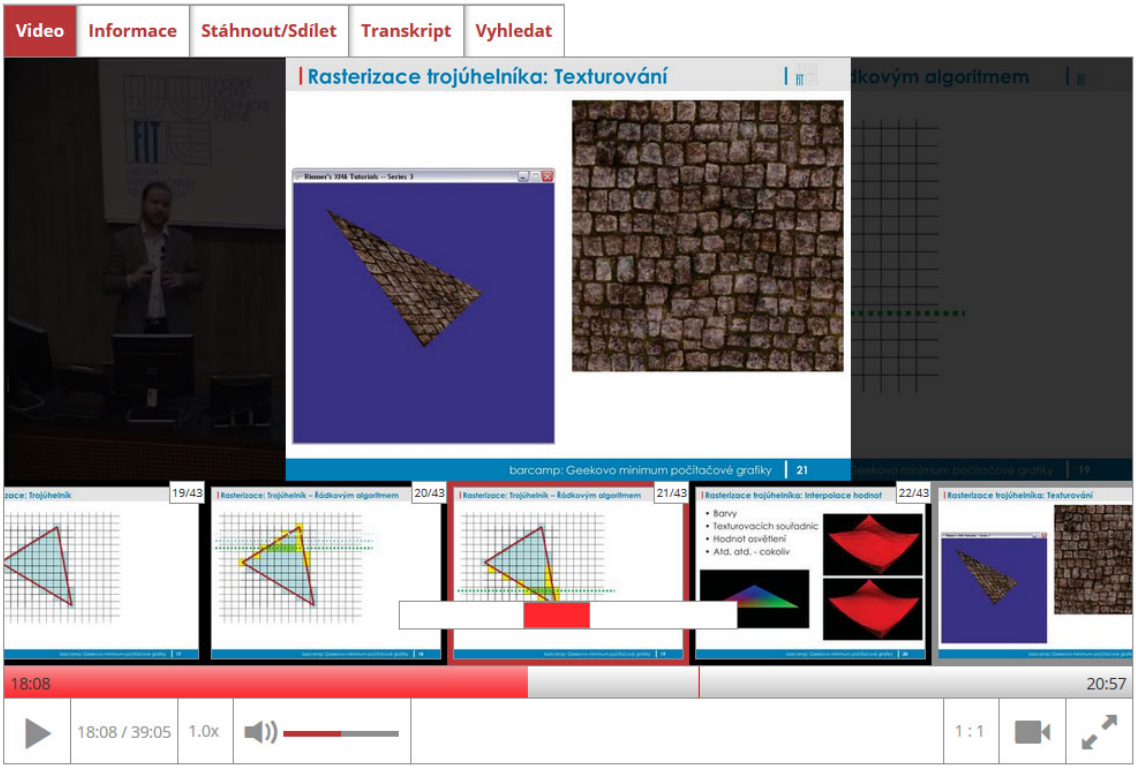
Pro klasická zařízení lze tento problém vyřešit pomocí detekce *hoveru* nad elementech *progress-baru*.

Na druhou stranu, na zařízeních dotykových je detekce jednodušší a stačí pouze reagovat na událost *scroll*.

4.5.5 Preview slajdu

Při události přejetí ukazatelem myši přes daný slajd v přehledu slajdů nabízí přehrávač možnost *preview* daného slajdu ve zvětšené podobě.

Preview je opět realizováno pomocí modálního elementu, který dočasně překrývá přehrávač videa a přehrávač slajdů. Pro dokreslení efektu náhledu navíc modální element způsobí "zatmění"prvků, které překrývá.



Obrázek 4.2: Vyobrazení *preview* slajdu

4.5.6 Synchronizace a Přetáčení

Objekt `SlidesPlayer` poskytuje tři různé možnosti přetáčení - přetáčení pomocí výběru slajdu, či dvousměrný posuv slajdu gestem prstu na dotykových zařízeních.

Všechny tři možnosti jsou řešeny prostým nastavením ukazatele aktivního slajdu seznamu slajdů na odpovídající hodnotu daného slajdu.

Tato funkcionalita je řešena ve funkcích `SlidesPlayer.rewindToSlideByID`, `SlidesPlayer.rewindToNextSlide` a `SlidesPlayer.rewindToPreviousSlide`. Po provedení přetočení je samozřejmě nutno zaslat signály objektům `VideoPlayer` a `Transcript` o *on-demand* synchronizaci pomocí standardních funkcí těchto dvou objektů `SlidesPlayer.syncToRewind`. Tato funkce již byla rozebrána pro objekt `VideoPlayer` v předchozí kapitole.

Objekt `SlidesPlayer` obsahuje rovněž vlastní implementaci této funkce, která je naopak volána při přetočení v rámci objektů `VideoPlayer` nebo `Transcript` a signalizuje objektu `SlidesPlayer` žádost o synchronizaci.

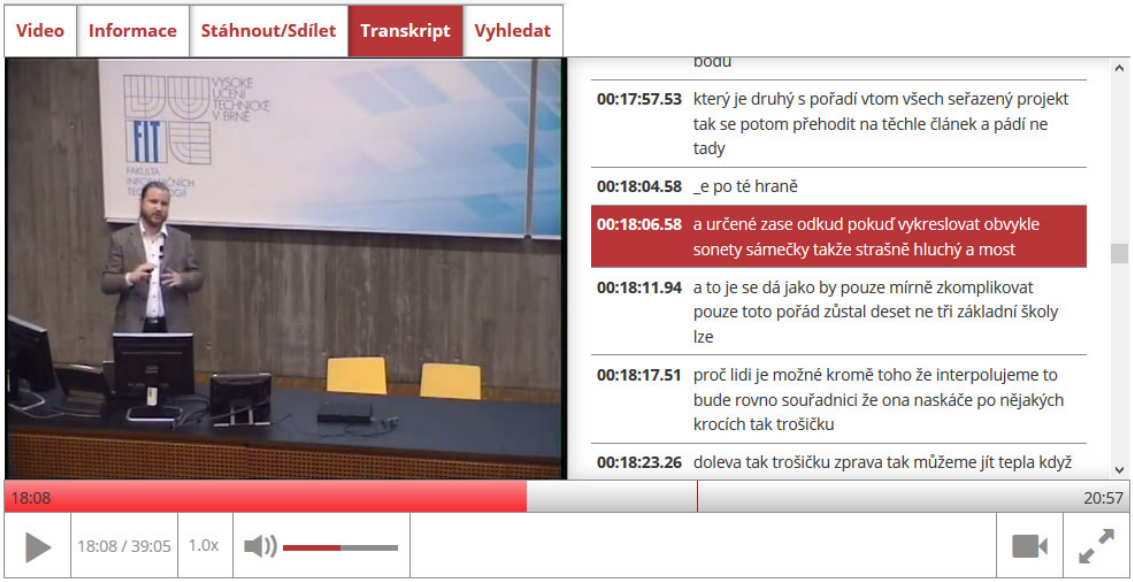
Funkce prochází seznam slajdů a hledá první slajd s časem vyšším, než je synchronizační čas, viz. následující úsek kódu.

```
$.each(this.slides, function(index, slide) {
   slideTime = slide['@attributes']['start'];

   if(slideTime >= time) {
       self.nextSlide = (index == 0) ? 0 : index-1;

       return false;
   }
   else if(index == self.lastSlide)
      self.nextSlide = self.lastSlide;
});
```

4.6 Transcript

Objekt `Transkript` je založený na totožných principech jako objekt `SlidesPlayer`. Rovněž tedy pracuje s jednosměrně vázaným seznamem, který ale v tomto případě není tvořen slajdy, nýbrž segmenty transkriptu. Veškerá manipulace se seznamem segmentů v rámci objektu `Transcript` je poté totožná s manipulací se seznamem slajdů. Z tohoto důvody nebudou implementační detaily tohoto objektu dále diskutovány.

Jediným rozdílem oproti objektu `SlidesPlayer` je fakt, že objekt `Transcript` musí před započetím výpočet odeslat asynchronní žádost na server Superlerlectures.com, odkud získá, pokud je transkript k dispozici, objekt obsahující jeho segmenty. Následně výsledek vhodně zformátuje a zobrazit tak, že nahradí běžnou pozici aktivního slajdu. Přehled slajdů v této záložce není zobrazen. Pokud transkript k dispozici není, objekt ukončuje svou činnost.



Obrázek 4.3: Vyobrazení transkriptu

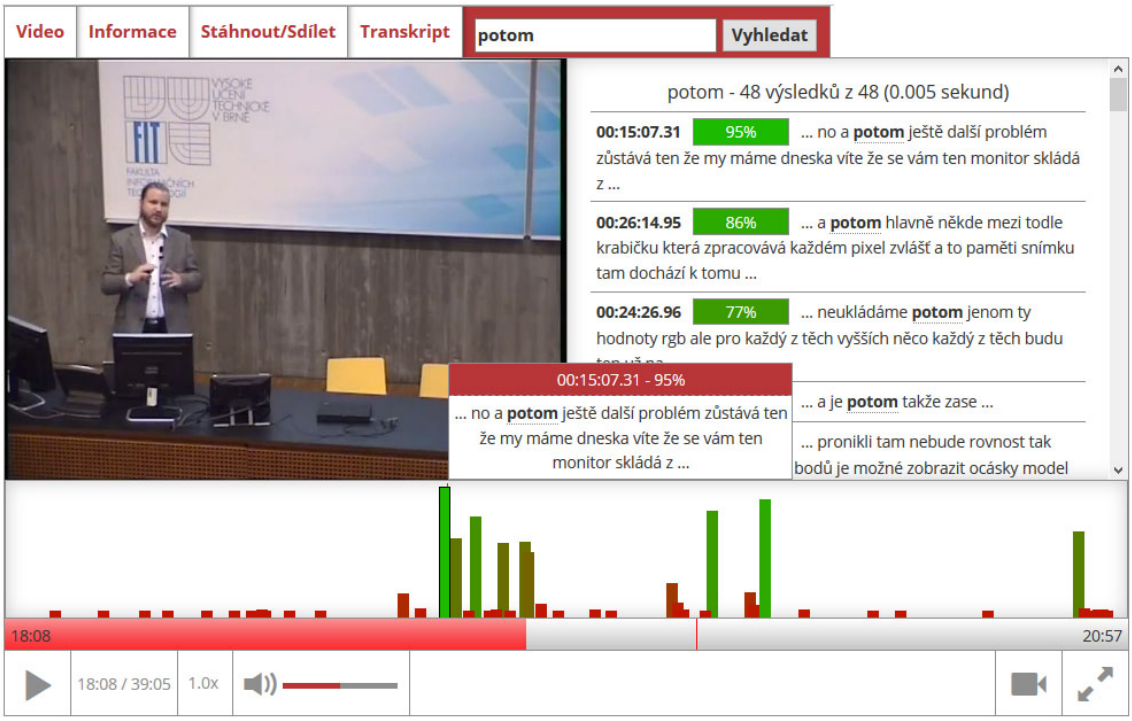
4.7 Search

Poslední záložku spravuje objekt `Search`.

4.7.1 Sestavení vyhledávání

Po výběru záložky Search je uživateli zobrazen jednoduchý formulář pro zadání vyhledávané fráze. Vyplněním fráze a odesláním formuláře uživatel potvrdí asynchronní dotaz na server Superlectures.com, odkud je získán objekt, obsahující, mimo jiné, časové známky všech výskytů dané fráze v audiu.

Časové známky jednotlivých výskytů fráze je dále možno využít k adresaci segmentů transkriptu. Pomocí volání funkce `Transcript.getSegmentByOccurrence` objekt `Search` získá pro každý výskyt fráze odpovídající segment, v němž vyznačí výskyt oné fráze. Výsledek vyhledávání pak formátuje dvěmi různými způsoby.



Obrázek 4.4: Vyobrazení vyhledávání (i pro konkrétní výskyt)

Výčet výskytů

Běžná pozice aktivního slajdu je v případě vyhledávání nahrazena výčtem výskytů, který posupně sestaví funkce `Search.buildTabOccurrence`. Pro každý výskyt je k dispozici časová známka segmentu transkriptu, ve kterém se výskyt nachází. Výběr daného segmentu rovněž funguje jako odkaz na daný čas záznamu. Po výběru se, opět, provádí veškerá nutná synchronizace, kterou řeší funkce `Player.rewindToTabOccurrence`. Za časovou známkou následuje, jak procentuální, tak barevný popis míry shody daného výskytu s frází. Posledním prvkem je pak textový výpis daného segmentu s vyznačením fráze.

Dotaz defaultně vrací prvních 10 výskytů v největší mírou shody. Uživatel může pomocí tlačítka *Zobrazit více*, na konci kontejneru výskytů, zažádat o další sadu výskytů.

Sloupcový graf výskytů

Přehled slajdů je rovněž nahrazen plochou vyhrazenou pro sloupcový graf výskytů, který sestaví funkce `Player.buildBarOccurrence`. Každý výskyt je vyobrazen, dle času, v odpovídající části časové osy (*progress-bar* videa slouží jako časová osa) a jak šířka, tak barva, daného sloupce popisuje míru shody výskytu s frází.

Z důvodu viditelnosti výskytů s nízkou mírou shody je nastavena bázová výška každého sloupce na 5% výšky kontejneru, míra shody pak pracuje se zbylými 95%.

Daný sloupec (na desktopových zařízeních) při události *hover* vystoupí do popředí. Tento přístup by měl zamezit určité míře nepřehlednosti zobrazení při velkém počtu výskytů a umožnit pohodlné procházení všech sloupců. Výstup do popředí je doprovázen zobrazením dalšího modálního elementu, který detailněji popisuje daný výskyt sloupcového grafu. Výběr daného výskytu opět funguje jako odkaz na daný čas záznamu, nicméně v tomto případě neodkáže na začátek segmentu, kde se výskyt nachází, ale přímo na daný výskyt fráze. Po výběru se, opět, provádí veškerá nutná synchronizace, kterou řeší funkce `Player.rewindToBarOccurrence`

Kapitola 5

Testování

Kapitola testování blíže specifikuje jak probíhalo testování ergonomie přehrávače nad množinou cílových uživatelů přehrávače Superlectures.com, stejně jako věnuje několik odstavců specifikaci testování kompatibility přehrávače na různých platformách, pod různými prohlížeči.

5.1 Ergonomie

Testování ergonomie přehrávače bylo provedeno dvěmi způsoby.

Prvním způsobem bylo iterativní testování přehrávače v průběhu implementace, pro které byla zvolena malá skupina respondentů o počtu čtyřech lidí. Žádný z těchto lidí nebyl technického zaměření a všichni by se dali označit jako průměrně technicky zdatní jedinci. Zmíněná skupina jedinců tímto způsobem poskytovala drobné informace, které byly již během implementace dané funkce použity k ladění.

Druhou fází testování pak bylo testování přehrávače jako celku. Této fáze se zúčastnilo právě 10 respondentů, kde každý respondent byl technické zaměření.

Pro tento účel byla vytvořena jednoduchá stránka, která simuluje zasazení nového přehrávače do webu Superlectures.com. Část z respondentů přistupovala k této stránce z mobilního zařízení a část z zařízení desktopového.

Pro každého respondenta byla připravena sada úloh, které měl vykonat. Obtížnost každé z úloh měl respondent po vykonání ohodnotit známkou (jako ve škole - 1 až 5) a nepovinně měl možnost přidat poznámku k danému úkolu. Na závěr pak dotazník obsahoval možnost ohodnocení celkové ergonomie přehrávače, nabídky funkcí přehrávače a designu přehrávače.

Seznam úkolů byl následující.

1. Otevřít web s přehrávačem
2. Vybrat přednášku *Geekovo minimum počítačové grafiky (A. Herout)*
3. Zjistit motivaci dané přednášky a informace o autorovi
4. Spustit záznam
5. Přetočit záznam na slajd, který se věnuje *Malířovu algoritmu*
6. Vyzkoušet chování přehrávače v režimu celé obrazovky.
7. Nahlédnout na přepis videa

8. Vyhledat v audiu frázi "malíř"a prozkoumat výsledky hledání
9. Stáhnout přepis audia ve formátu *epub*
10. Sdílet záznam na zvolené sociální síti.
11. Prozkoumat další možnosti přehrávače.

Vyhodnocením dotazníků a přiložených poznámek byla extrahována následující fakta.

- Žádný z úkolů nepředstavoval pro uživatele problém. Uživatelské rozhraní přehrávače bylo více, než polovinou uživatelů spontánně označeno v rámci poznámek za intuitivní.
- Uživatelé ocenili funkci pro úpravu rychlosti přehrávání záznamu, stejně jako funkci pro změnu poměru videa ke slajdům. Nicméně se objevil názor, že by bylo vhodné do volby těchto funkcí zavést větší stupeň flexibility v podobě nahrazení *selectboxu* ovládacím prvkem typu *slider*. [Tento požadavek bude zařazen do dalšího vývoje]
- Objevil se názor, že by bylo vhodné do přehrávače zavést dodatečné ovládání s využitím klávesových zkratek a to hlavně pro funkce *Play/Pause*, úplně ztlumení audiozáznamu, přechod do režimu celé obrazovky a dokonce i změnu záložek. [Tento požadavek bude prodiskutován a eventuálně zařazen do dalšího vývoje]
- Uživatel velmi ocenili funkce náhledu na slajdy a transkriptu, které provádí automatické centrování k aktuálnímu času videa. [Tento požadavek bude zařazen do dalšího vývoje]
- Část respondentů, která přistupovala k přehrávači z desktopového zařízení by ocenila možnost přetáčení přehledu slajdů pomocí kolečka myši. [Tento požadavek bude zařazen do dalšího vývoje]
- Téměř všichni uživatelé, přistupující z mobilních zařízení, přímo ocenili vyobrazení přehrávače v režimu celé obrazovky.
- Rovněž téměř všichni uživatelé ocenili možnost přetáčení v režimu celé obrazovky pomocí dotyku.
- Několik uživatelů ocenilo možnost vložení záznamu do vlastních osobních stránek v podobě *embeded* prvku.
- Téměř všichni uživatelé přímo ocenili vyobrazení záložky pro vyhledávání a, především pak přehledné odlišení jednotlivých výskytů na časové v podobě sloupcového grafu.

Hodnocení obtížnosti jednotlivých úloh dopadlo následovně. První hodnota udává průměrnou známku a v závorce následuje odpovídající rozdělení hodnot.

1. 1 (1 - 100%)
2. 1 (1 - 100%)
3. 1,2 (1 - 80%, 2 - 20%)
4. 1 (1 - 100%)
5. 1,2 (1 - 80%, 2 - 20%)
6. 1,1 (1 - 90%, 2 - 10%)
7. 1 (1 - 100%)
8. 1,2 (1 - 80%, 2 - 20%)
9. 1,4 (1 - 60%, 2 - 40%)
10. 1,1 (1 - 90%, 2 - 10%)
11. Nehodnoceno

Hodnocení ergonomie, funkcí a designu pak následovně.

- Ergonomie přehrávače - 1,3 - (1 - 70%, 2 - 30%)
- Funkce přehrávače - 1,5 - (1 - 50%, 2 - 50%)
- Design - 1,6 - (1 - 50%, 2 - 40%, 3 - 10%)

Díky iterativnímu testování se podařilo odhalit mnoho drobných nedostatků, které vznikaly, již v vývoje konkrétních funkcí přehrávače. Tyto nedostatky byly, za běhu, opraveny, resp. dolazeny.

Poznámky v závěrečném dotazníku odhalily několik dalších nedostatků, resp. možných rozšíření, přehrávače do budoucna. Tyto změny budou diskutovány a případně implementovány nad rámec této práce.

Z výsledků obtížnosti lze vypozorovat, že se známky drží, ve všech případech, mezi hodnotami 1 a 2, což znamená že obtížnost základních úloh manipulace s přehrávačem je velmi nízká a, dle poznámek, funkcionalita je zároveň uspokojující, což bylo jedním z cílů této práce.

Co se týče závěrečného hodnocení, dopadl "nejhůře"design přehrávače. Pro změny designu je zde nicméně spousta prostoru a samotný design bude záležet na konkrétní konferenci, pro kterou přehrávač využit.

5.2 Kompatibilita

Velká pozornost byla rovněž věnována testování kompatibility všech funkcí na různých kombinacích zařízení - prohlížeč, tedy cílových platformách. Dále bylo důkladně testováno chování *layoutu*, které má mnoho dynamických prvků a proto je nutno rovněž testovat chování zobrazení na různých cílových zařízeních.

I toto testování bylo prováděno v iteracích. Každou iteraci, opět, tvořila nově implementovaní funkce, resp. významná změna *layoutu* přehrávače.

Cílová zařízení, která byla použita pro testování, společně s odpovídajícími prohlížeči jsou.

- Klasický desktopový systém pod (Windows)
  - Mozilla Firefox 46.0.1
  - Google Chrome 51.0.2704.54
  - Internet Explorer 11
- Samsung Galaxy Tab A a Samsung Galaxy S3 (Android)
  - Opera Mini
  - Firefox for Android
  - Web Explorer
  - Chrome for Android
  - Nativní Android prohlížeč
- Apple iPad, 3. generace (iOS)
  - Nativní iOS Safari

Během testování byla objevena celá řada bych, jak v chování layoutu, tak v chování funkcí přehrávače. Tato chyby souvisely, převážně, se zpracováním dotykových událostí. Všechny odhalené chyby byly následně opraveny.

Jedinou chybou, která přetrvala je nedostatek v nativním prohlížeči Android, který na některých zařízeních (v tomto případě Samsung Galaxy Tab A) korektně nereaguje na události *swipe* při této kombinaci zařízení-prohlížeč není možno provádět přetáčení o jeden slajd vpřed/vzad. Tento nedostatek bude zařazen do požadavků, týkajících se dalšího vývoje přehrávače Superlectures.com.

Kapitola 6

Závěr

V rámci této práce byl implementován nový přehrávač pro web Superlecture.com, který by měl sloužit jako náhrada přehrávače stávajícího. Přehrávač byl implementován s ohledem na požadavek, aby bylo možné jej užít stylem *plug and play*. Do přehrávače se podařilo zapouzdřit naprostou většinu funkcí, které poskytuje přehrávač aktuální, což bylo ostatně jádrem zadání. Tato množina byla rovněž rozšířena o řadu funkcí, které lze klasifikovat jako nadstavbu přehrávače aktuálního. Přehrávač byl rovněž implementován s velkým důrazem na přehlednou a robustní strukturu, která by měla usnadnit další vývoj a napojování dalších rozšiřujících funkcí do přehrávače. Velký důraz byl při implementaci kladen i na rychlost přehrávače. Přehrávač byl implementován tak, aby jej bylo možno pohodlně využít jak na zařízeních desktopových, tak na zařízeních mobilních.

Během závěrečných testů přehrávače bylo zjištěno, že je poptávka po několika jednoduchých rozšířeních, které mohou být implementovány v rámci budoucího vývoje přehrávače. Struktura přehrávače by implementaci, jak těchto poptávaných funkcí, tak funkcí, které se objeví mezi požadavky budoucími, částečně usnadnit. Nicméně jakýkoli další postup spadá mezi rozhodnutí správců služby Superlectures.com.

Literatura

[1] HTML 5 - A vocabulary and associated APIs for HTML and XHTML. http://www.w3.org/TR/html5/, accessed: 2016-1-8.

[2] The CSS standardization process - Levels 1, 2, 3 and above. https://www.w3.org/Style/2011/CSS-process, 2011-11-14 [cit. 2016-5-3].

[3] ECMAScript® Language Specification. http://www.ecma-international.org/ecma-262/5.1/#sec-10, 2011 [cit. 2016-5-3].

[4] Bender, J.: Tap vs. Click: Death by Ignorance. https://coderwall.com/p/bdxjzg/tap-vs-click-death-by-ignorance, 2012-7 [cit. 2016-5-3].

[5] Doktorova, L.: doT.js origins. http://olado.github.io/doT/e, 2014-12-2 [cit. 2016-5-3].

[6] jQuery foundation, T.: jQuery Plugins. http://learn.jquery.com/plugins/, 2015-3 [cit. 2016-5-3].

[7] jQuery foundation, T.: jQueryUI. http://jqueryui.com/about/, 2016 [cit. 2016-5-3].

[8] Goodwin, R.: What is CSS3? http://www.sitepoint.com/web-foundations/what-is-css3/, 2012-10-8 [cit. 2016-5-3].

[9] Internationala, E.: Standard ECMA-404: The JSON data interchange format. http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf, 2013.

[10] Jelínek, L.: jQuery (16) – úvod do jQuery UI. http://www.linuxexpres.cz/software/jquery-16-uvod-do-jquery-ui, 2015-8-7 [cit. 2016-5-3].

[11] Network, M. D.: Touch events. https://developer.mozilla.org/en-US/docs/Web/API/Touch_events, 2016-1-22 [cit. 2016-5-3].

[12] Network, M. D.: Vendor Prefix. https://developer.mozilla.org/en-US/docs/Glossary/Vendor_Prefix, 2016-3-3 [cit. 2016-5-3].

[13] W3C: W3C - Touch events. https://www.w3.org/TR/touch-events/, 2013-9-10 [cit. 2016-5-3].

Přílohy

Seznam příloh

A Objekt Lecture **52**

B jQuery Boilerplate **54**

C Konstrukce doT.js **55**

Příloha A

Objekt Lecture

```
<lecture>
    <@attributes>
      <id>18</id>
      <audio_search_id>main2</audio_search_id>
      <category_id>100</category_id>
      <name>Geekovo minimum počítačové grafiky</name>
      <language>cs</language>
      <url>https://www.superlectures.com/barcampbrno2011/</url>
    </@attributes>
    <authors>
      <person>
        <@attributes>
          <name>Adam Herout (www.herout.net)</name>
        </@attributes>
        <twitter>
          <@attributes>
            <src></src>
            <title></title>
          </@attributes>
        </twitter>
      </person>
    </authors>
    <description>Anotace</description>
    <recorded>2011-10-15 09:30 - 10:15, Main Hall</recorded>
    <datetime_recorded>2011-10-15 09:30:00</datetime_recorded>
    <datetime_added>2011-10-25 14:54:00</datetime_added>
    <thumbnail>
    <thumbnail_large>
      <@attributes>
        <src>https://www.superlectures.com/</src>
      </@attributes>
    </thumbnail_large>
    <video></video>
    <audio>
      <@attributes>
```

```
      <duration>2324</duration>
    </@attributes>
    <audio_mp3>
      <@attributes>
        <src>https://www.superlectures.com/</src>
        <download_src>https://www.superlectures.com/</download_src>
        <size>13949577</size>
      </@attributes>
    </audio_mp3>
  </audio>
  <subtitles></subtitles>
  <slides>
    <@attributes>
      <number_of_slides>43</number_of_slides>
    </@attributes>
  </slides>
  <pdf>
    <@attributes>
      <src>URLURL</src>
      <size>2294036</size>
    </@attributes>
  </pdf>
  <link>
    <@attributes>
      <src></src>
    </@attributes>
  </link>
  <views>719</views>
  <stats>
    <@attributes>
      <url>https://www.superlectures.com/</url>
    </@attributes>
    <description>Popis videa</description>
    <views>228</views>
    <well_replayed>89.64</well_replayed>
    <plays>387</plays>
    <views_img></views_img>
    <histogram_img></histogram_img>
  </stats>
  <search>
    <@attributes>
      <enabled>true</enabled>
    </@attributes>
  </search>
</lecture>
```

Příloha B

jQuery Boilerplate

```
;(function($, window, document, undefined) {

   "use strict";

   var pluginName = "slplayer",
      defaults = {
         //Defaultni parametry prehravace
      };

   var Plugin = function(element, options) {
      this.element = element;

      this.settings = $.extend({}, defaults, options);
      this.init();
   }

   $.extend(Plugin.prototype, {
      init: function() {
         ;
      });
   };

   $.fn[pluginName] = function(options) {
      return this.each(function() {
         if(!$.data(this, "plugin_" + pluginName)) {
            $.data(this, "plugin_" + pluginName, new Plugin(this, options));
         }
      });
   };
})(jQuery, window, document);
```

Příloha C

Konstrukce doT.js

Zápis zajišťující podmíněné vykreslení záložky *Transcript*.

```
{{? it.transcript}}
   <li class="tab">
      <a href="#transcript">
         {{=it.tabs.transcript}}
      </a>
   </li>
{{?}}
```

Zápis iterace přes atributy objektu, resp. pole.

```
{{~ it.info.authors :a }}
   <div class="author">
      <span class="name">
      {{ for(var p in a) { }}
         {{? p != '@attributes' }}
            <a class="{{=a[p].title}}" href="{{=a[p].src}}">
               <img title="{{=a[p].src}}" src="{{=a[p].icon}}"/>
            </a>
         {{?}}
       {{ } }}
       </span>
   </div>
{{~}}
```

Tato komplexnější struktura je vyňata přímo z šablony přehrávače. Je použita ve štítku Informace, kde v rámci vnějšího cyklu iteruje přes všechny autory dané přednášky a v rámci cyklu vnořeného, rovněž, přes konkrétní způsoby kontaktu autora, které takto vypisuje do přehledné tabulky.